5-1-2002

Analysis of H/W & S/W techniques for data reduction in high speed digital image processing

Paul DeSanctis

Analysis of H/W & S/W Techniques For Data Reduction in High Speed Digital Image Processing

by

Paul A. DeSanctis

A Thesis Submitted

in

Partial Fulfillment of the

Requirements for the Degree of

MASTERS OF SCIENCE

in

Computer Engineering

Approved by:

Committee Principle: _____ Date: 7-25-2002

Dr. Muhammad Shaaban, Ph.D
Assistant Professor

Committee Member: _____ Date: May 24, 2002

Dr. Roy S. Czernikowski, Ph.D
Professor

Committee Member: _____ Date: 5/24/02

Pawel Sniatala,Ph.D
Assistant Professor

Department of Computer Engineering

Kate Gleason College of Engineering

Rochester Institute of Technology

Rochester New York, USA

May 2002

THESIS RELEASE PERMISSION FORM

Rochester Institute of Technology

Analysis of H/W & S/W Techniques For Data Reduction in High Speed Digital Image Processing

I, Paul A. DeSanctis, hereby grant permission to any individual or organization to reproduce this thesis in whole or in part for non-commercial and non-profit purposes only.

Paul A. DeSanctis

8/8/2002

Date

Acknowledgements

To my wife Teresa, you have supported me during this educational milestone. Thank you for the understanding and care of our children. Without your support and understanding this would not have been possible.

To my four children, Amanda, Natalie, Vincent, and Dominic. Your watchful eyes inspired and motivated me to complete my degree. I hope that in the future you will follow my example and patiently work to achieve your goals.

Abstract

With the widespread utilization of charge-coupled-devices, there is much interest in methods to efficiently process images. The processing, manipulation, and storage of photographic quality digital images place significant demands on today's computers. Even with today's high performance bus structure and real-time operating systems, manipulating full resolution image data may quickly overwhelm computer hardware and software. In response to this, data reduction techniques have been developed to aid in resolving this problem. Two common data reduction techniques include data sub-sampling and data averaging.

Data sub-sampling approach is simplistic in nature and perhaps easiest to implement in both hardware and/or software. This approach involves sub-sampling the full resolution image data to a lower resolution. Selection of sub-sampled element of the full resolution image is random in nature. This random selection makes sub-sampling an effective technique for flat image fields but degrades or softens the image for edges information quality/content.

Data averaging approach is more difficult and complex to implement in both hardware and software than the sub-sampling approach. The data averaging approach involves a two-dimensional averaging function to sub-sample the full resolution image data to a lower resolution. Averaging area parameters may be chosen to average X consecutive pixels, and Y consecutive lines. Although more complex, data averaging more effectively retains edge information.

This thesis investigates the two-dimensional, pixel data-averaging method for data reduction. It supports the use of a pixel-averaging algorithm in conjunction with, or independent from compression techniques which may be employed elsewhere within the same system. Hardware and software implementations are presented to solve this system problem. The hardware architecture design is based on a pixel averaging application specific integrated circuit. Software routines written in C programming language are presented to

perform this data-averaging task. Performance comparisons are made between the hardware and software implementations for image resolutions up to 2048 by 3072 pixels, and under several averaging conditions.

This thesis also provides a survey of various types of charge-coupled devices sensors, focusing on their abilities and limitations for data averaging. It presents several applications where this type of data reduction would be advantageous.

Table of Contents

List of Figures

List of Tables

List of Equations

Glossary of Terms

A/D – Analog to Digital Conversion,

The process of converting an analog voltage signal level to a digital representation

ALU – Arithmetic Logical Unit,

A circuit that is capable of performing arithmetic, logical, and data manipulation operations on binary numbers

ASIC - Application Specific Integrated Circuit,

A semi-custom or full-custom integrated circuit, used for integrating large amounts of digital logic functions.

Binning or Super Pixel,

On chip noiseless summation of charge stored in electrodes, before it is read into the output amplifier. Results in an increased sensitivity at the expense of spatial resolution. Off chip binning (or data averaging) is a method of reducing data that involves computing a two-dimensional average value of a pixel neighborhood, for generation of a lower resolution image.

CCD – Charge Coupled Device,

An analog integrated circuit that uses an array of electrodes to convert photons of light into a proportional electrical charge. The intensity of this charge relates to a color in the color spectrum.

CFA - Color Filter Array,

Color filters designed to act as band pass filters for light. Depending on their construction, they reflect or absorb specific wavelength bands.

CTE – Charge Transfer Efficiency,

The fraction of electrons passed to the next photo site position during the CCD readout process.

CPU – Central Processing Unit,

Reads instructions from data and memory and performs operations on them. Exchanges information with the user through some input/output interface.

FIFO - First-in-first-out memory device,

A memory queuing storage device where the oldest data element is read out first.

FF CCD - Full Frame CCD Sensor Device,

A type of area CCD, which is one of the easiest CCD devices to fabricate and operate.

FT CCD - Frame Transfer CCD Sensor Device,

A type of area CCD most commonly used for scientific imaging, which has an imaging and blanked storage section.

H/W – Hardware,

Electronic circuitry that is designed to perform specific functions.

Pixel - Picture element,

The contraction of two words: picture and element. A pixel represents a single sampling point of an image.

LSB – Least Significant Bit,

The right most number in a binary number field.

MSB – Most Significant Bit,

The left most number in a binary number field.

QE – Quantum Efficiency,

Specifies the percentage of photons incident on a CCD device that are detected.

SNR - Signal to Noise Ratio,

The relative magnitude of the signal compared to the uncertainty in that signal on a per pixel basis.

S/W – Software,

Programs or applications that run on H/W to perform a specific task.

VHDL – Very High Speed Integrated Circuit Hardware Descriptive Language,

A hardware descriptive language used in the definition and specification of electronic circuits.

VLSI – Very Large Scale Integration,

A term used to describe semiconductor circuits composed of thousands of logic cells or memory elements.

Chapter 1
Introduction & Background

The processing, storage, and manipulation of photographic quality digital images place significant demands on today's computers. Even with today's high performance bus structure and real-time operating systems, unnecessarily manipulating full resolution image data could overwhelm a computer's hardware and software.

In response to this, data reduction techniques have been developed to assist in resolving this problem. Two common data reduction techniques include data sub-sampling, and data averaging.

System Engineers do comparative analysis of software versus hardware solutions to these types of engineering problems. They assess the impact on overall system performance by resolving these problems in either hardware (H/W) or software (S/W). S/W is often easier and quicker to implement, allows for changes in the future, but may utilize precious system CPU bandwidth. Custom H/W has a longer development cycle and is more difficult to modify its implementation. H/W solutions utilize minimal CPU bandwidth and have the potential to operate significantly faster.

This thesis will discuss data averaging concepts and applications. It supports the use of pixel averaging algorithms in conjunction with other data compression protocols. H/W and S/W techniques are investigated to solve this system problem.

1.1 Introduction

The data averaging approach is more difficult and complex to implement in both hardware and software than the data sub-sampling approach. Increased image quality in the processed images justifies the additional complexity introduced by two-dimensional data averaging.

Data averaging involves two-dimensional averaging of a 'pixel neighborhood' to sub-sample the full resolution image to a lower resolution. Area averaging parameters may be chosen to average a specific number of consecutive pixel columns by a specific number of consecutive pixel rows. Although more complex, data averaging is very effective and retains more edge information. Averaging hardware and software is more complex, since it is now necessary to incorporate an intermediate sum FIFO or buffer. In most cases where image quality of the sub-sampled image is important, the data-averaging approach is chosen over data sub-sampling.

Technological advances in VLSI technology make it possible to incorporate the image data sub-sampling hardware into an Application Specific Integrated Circuit (ASIC). These circuitry advances allow for faster clocking frequencies. In addition, data pipelining techniques may be utilized in order to make this hardware approach even more efficient.

Several variables impact the efficiency and processing speed of the software approach. Some of these variables include: software implementation and CPU platform type. With advances in CPU speeds and architectures, it is becoming more common for the main system CPU to perform many of these imaging operations in S/W.

1.2 Background

Images may be recorded digitally utilizing multiple image resolutions based on use. Consider the following example to illustrate this need for multiple resolution levels of low, medium, and high. This example is summarized in Table 1-1.

Image Quality Class	Typical Resolution	Typical Application
Low Resolution	192 x 128	Thumbnail image
Medium Resolution	1024 x 1536	Graphics display
High Resolution	2048 x 3072	Photographic prints and enlargements

Table 1-1: Multiple Resolution Application Example

A thumbnail image is a small, postage-stamp size representation of an image, which would utilize a low resolution. Typically, a thumbnail-sized image is used for display on the screen when an image file is selected for opening. A thumbnail resolution image is also used for printing on an index print, which is a single sheet representation of an order of photographic service prints utilizing thumbnail-sized images.

Graphic display terminals require the display of images at a medium resolution. Technical limitations of the terminal screen, along with video subsystem performance limit the ability to display high-resolution images. Medium-level resolution images have the added advantage of reducing the workload on the system or CPU.

High-resolution images are needed for applications, such as medical imaging, digital negative storage, and high-resolution photographic printing. Digital negatives are used instead of film-based negatives to make reprints or enlargements of a scanned image. In some photo-finishing systems, digital negatives are stored electronically in archival and retrieval systems so that customers may order reprints without bringing in the photographic negative strip. Photographic printers utilize high-resolution data to generate high quality output images. Typically, a photographic printer may utilize this high resolution image for output print sizes ranging from 3R (3.5" x 5.0") service prints to enlargement prints up to 11" x 14" in size. The print resolution varies based on both the scanned resolution and the output print size. Typically, the minimum acceptable photographic printing resolution is 200 dpi

(dots per inch). Several home inkjet printers support photographic quality of up to 600 dpi or greater.

1.3 Data Averaging Applications

Several digital imaging fields application areas could benefit from data averaging. Any field that involves scanning, transmission, storage, and analysis of digital images could benefit from this technology. Example applications include satellite imaging, scan to print applications, machine vision, astronomy, and any imaging application with the need for increased frame rates.

1.4 Thesis Limitations

There are several techniques, standards and protocols used in the data compression field. The discussion of these compression standards is outside the scope of this study. This thesis work discusses and implements H/W & S/W averaging algorithms to produce smaller resolution images; however, compression algorithms could be utilized to further compress averaged images.

The S/W findings in this work are based on data averaging programs written in the C programming language without significant optimization. Timing benchmarks of these functions are for a typical desktop PC platform running under the Windows 2000 environment. Key functions or code segments could have been written in assembler or utilizing multi-media instructions (MMX). The following sections present findings based on standard coding practices, without utilizing these code optimization techniques.

1.5 Significant Findings

This work describes system-scanning issues when processing large amounts of data. It details several CCD sensor formats for various applications, addresses tradeoffs, advantages,

and disadvantages of several scanning formats, and details CCD specification parameters that impact performance.

A synchronous, highly pipelined data reduction H/W architecture design for a tri-linear CCD imaging system is presented. This architecture processes 12-bit color interleaved pixels at several input and output resolutions, consists of an ASIC, and supporting memory storage devices, and is scalable, supporting a broad range of input and output imaging resolutions. Averaging mode, input or output resolutions does not impact performance of this H/W architecture. Regardless of these parameters, the performance remains constant, introducing only 6-clock cycles of latency to the data pipeline.

The ASIC is developed utilizing Esclade development tool environment that allows for flexible input description methods including: schematic, state, truth table, and VHDL. The output format is VHDL code that is portable and allows many vendors' silicon to be evaluated. The selection of the actual target device or vendor could be based on which device best meets design, performance, and cost requirements.

Synplicity synthesis tools were utilized and three Altera device families were selected. This device operated at a maximum frequency of 120 Mhz.

The S/W programs support two-dimensional pixel averaging at user specified input and output resolutions. This program outputs the elapsed execution time for the selected resolutions. Findings detailed here conclude a significant timesaving by utilizing averaging to reduce the amount of data transferred. An upper bound of 770 milliseconds to copy an image 2048 x 3072 image. When the same image was averaged in mode 4 (16 x 16 area), this execution time decreased to 150 milliseconds.

The shared PCI bus is more heavily utilized in S/W implementation versus the H/W implementation. In some systems, this could present an additional problem due the increased competition in obtaining ownership of the PCI bus.

1.6 Thesis Outline

The following chapters will further detail the issues on H/W & S/W Techniques for Data Reduction in High Speed Digital Image Processing. Chapter 2 is a primer on scanning technology, various CCD sensor formats, on and off chip CCD binning capabilities, and binning applications. Chapter 3 presents the H/W architecture design and ASIC description. Chapter 4 presents the S/W architecture and code design description. Chapter 5 presents the H/W and S/W simulation results. Chapter 6 presents conclusion remarks.

Chapter 2
Scanning Technology & CCD Terms

The primary function of a CCD device is to convert light (photons) into an electronic charge [9]. There are several types of scanning techniques utilized to accomplish this task.

This chapter provides background information on Scanning Technology. It discusses several scanning methods, their advantages, and disadvantages. This chapter also introduces CCD terminology, and present important CCD specifications that impact scanning performance. CCD noise sources and techniques for minimizing them are detailed. Pixel binning concept for data reduction and improved signal to noise performance are presented. Additionally, this chapter briefly reviews current applications for which data reduction from pixel binning is advantageous.

2.1 Introduction to Scanning Techniques & CCD Sensor Types

The CCD device serves as an "electronic eye", capturing light and converting it to a charge [5]. CCD techniques and principles are common in all sensor formats. This CCD process may be simplified to the following 3 steps:

1) An exposure cycle converts light into an electrical charge, which is held within the CCD photo sites.

2) A charge transfer cycle which transports the charge around the CCD.

3) A voltage conversion cycle that measures the charge present at the output amplifier and converts it to a voltage level.

Scanners utilize CCD devices to capture digital images. Scanner applications include the following: medical imaging, document imaging, film imaging, and space-borne imaging systems.

Three CCD sensor formats are available for accomplishing scanning tasks: point, linear, and area scanning. The three CCD scanning formats are shown in Figure 2-1 [1].

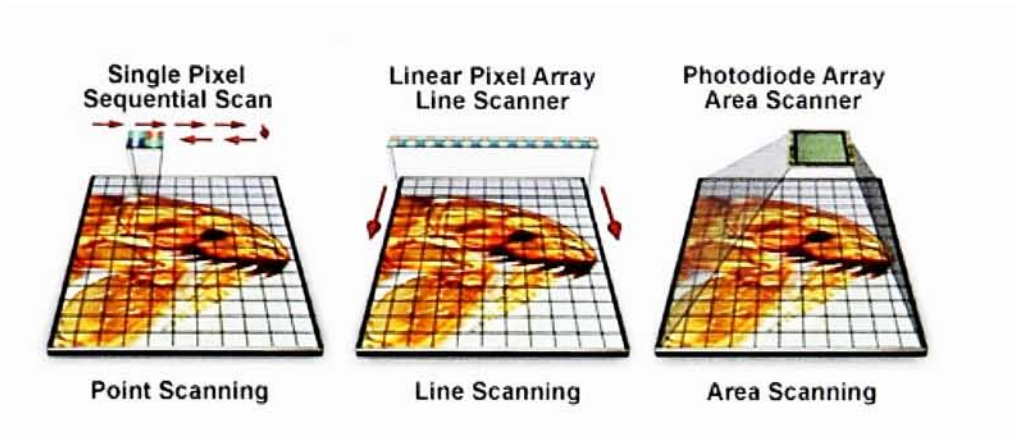

Figure 2-1: CCD Scanning Formats

Motion is necessary to capture the image in point and line scanning. Either the CCD sensor or media being scanned may be moved to accomplish this task. Area scanning is usually accomplished when both media and sensor are stationary. The following sections provide additional detail of each of these scanning formats.

2.1.1 Point Sensors

A single picture element (pixel) sequential scan involves capturing of each pixel, one at a time. Although this approach is simplistic, registration is very difficult to maintain. This imposes rigorous mechanical constraints on these types of systems. Registration errors cause degradation in the image quality. Additionally, the productivity of this scanning system is very low [2].

2.1.2 Linear Sensors

Linear array CCDs use multiple point pixels arranged to form a line of light sensitive silicon. The linear approach resolves one direction of the mechanical tolerance and

registration issues of the point scanning method. Figure 2-2 is a physical cross section of a linear CCD sensor device [3].

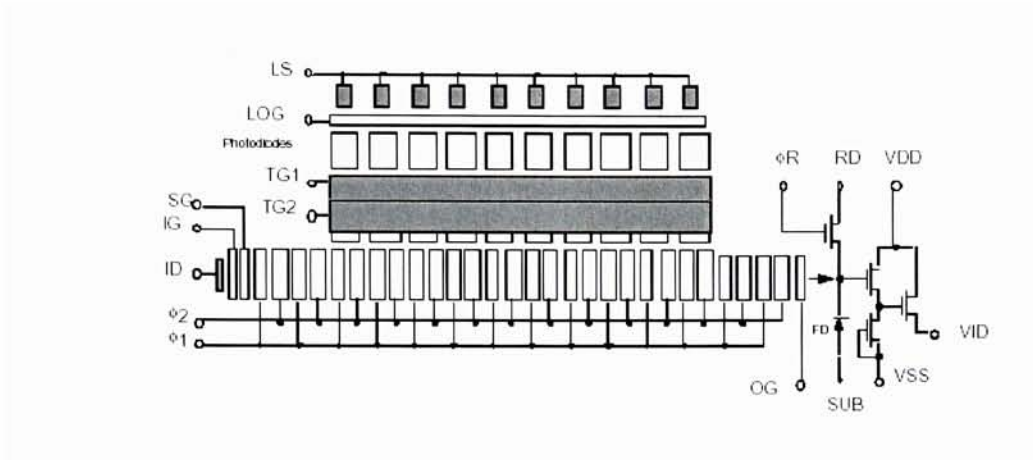

Figure 2-2: Linear CCD Sensor

Linear CCDs are usually monochrome. Capturing red, green and blue color information often requires three separate exposure and read out cycles. In addition to successive exposures, a color filter wheel is often utilized. The position of the filter wheel would be changed between each successive CCD exposure to light. Such a system is shown in Figure 2-3 [2]. Pixel data from a linear CCD is transmitted in color planar form, due to the sequential exposure cycle.

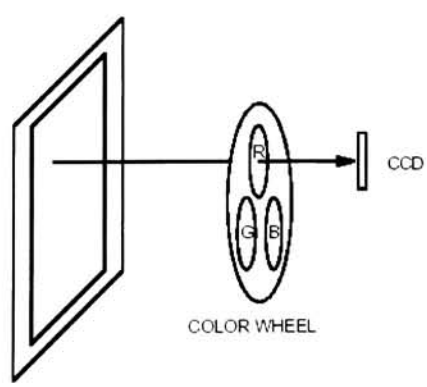

Figure 2-3: Sequential Capture Linear CCD System

Tri-linear CCDs eliminate the difficulties of sequential exposure cycle, and the need for a color filter wheel. Three separate linear CCDs are aligned and coated with integral color filter array (CFA) filters [9], allowing for the previously off sensor filtering to happen on the CCD sensor. CFA's are designed to act as band pass filters that reflect or absorb specific wavelength bands. Tri-linear CCDs makes each of the three linear arrays sensitive to only a single color. Figure 2-4 is a physical cross section of a tri-linear CCD device [4].

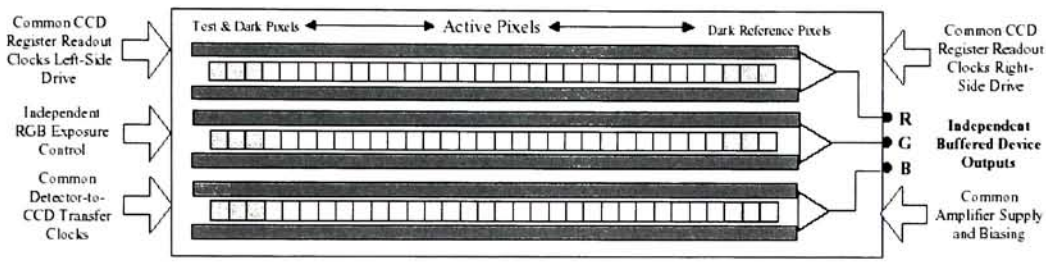

Figure 2-4: Tri-Linear CCD Sensor

In a tri-linear CCD, pixel data is often captured for the red, green and blue color channels simultaneously, and digitized. Often in tri-linear CCDs, the three separate color channels are merged to a single color interleaved channel after digitization. This is very cost effective, in that it reduces the number of interconnections for routing pixel data. The following is an example of color interleaved pixel data format, on this merged channel:

Red pixel # 1, green pixel # 1, blue pixel # 1, red pixel # 2, green pixel # 2, blue pixel # 2, ...

2.1.3 Area CCD Sensors

There are many types of area CCD sensors. An area CCD is a 2 dimensional array of light sensitive silicon. The three most common types of area CCD sensors are full frame (FF) frame transfer (FT) and interline devices.

The FF CCD is one of the easiest CCD devices to fabricate and operate [2]. It consists of vertically arranged parallel shift registers for exposure charge integration and transport. Figure 2-5 shows a cross section of a FF CCD device [3].

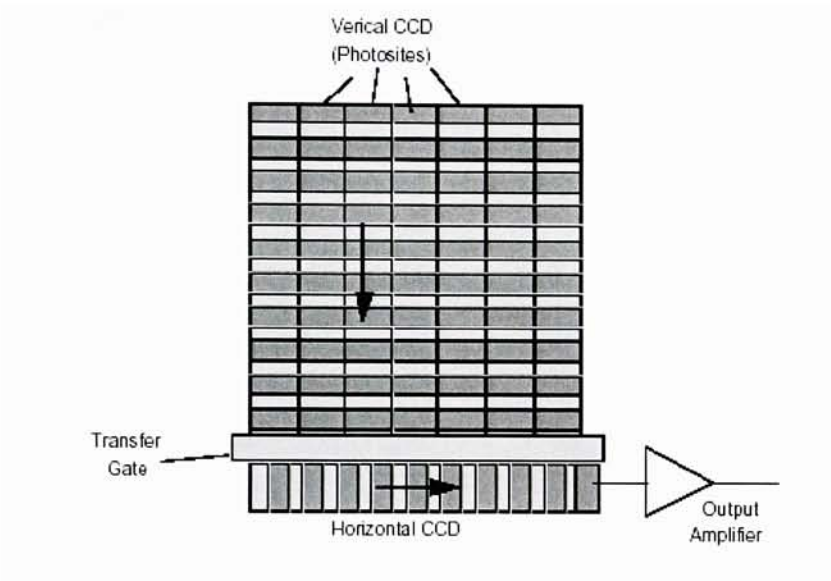

Figure 2-5: Full Frame CCD Sensor

The scene to be recorded is projected on the FF CCD. The vertical (or parallel) photo sites in a FF CCD are used for both image acquisition and read out. One row of data is transferred from the vertical photo sites into the horizontal (serial) read out register which then clocks the entire row of data to the charge sensing output amplifier. After the horizontal pixel data for that row has been completely clocked out, the next row of data is vertically clocked into the horizontal shift register. This process repeats until the entire image is transferred from the full frame CCD. This normal clocking technique for a full frame CCD is illustrated in Figure 2-6 [18].

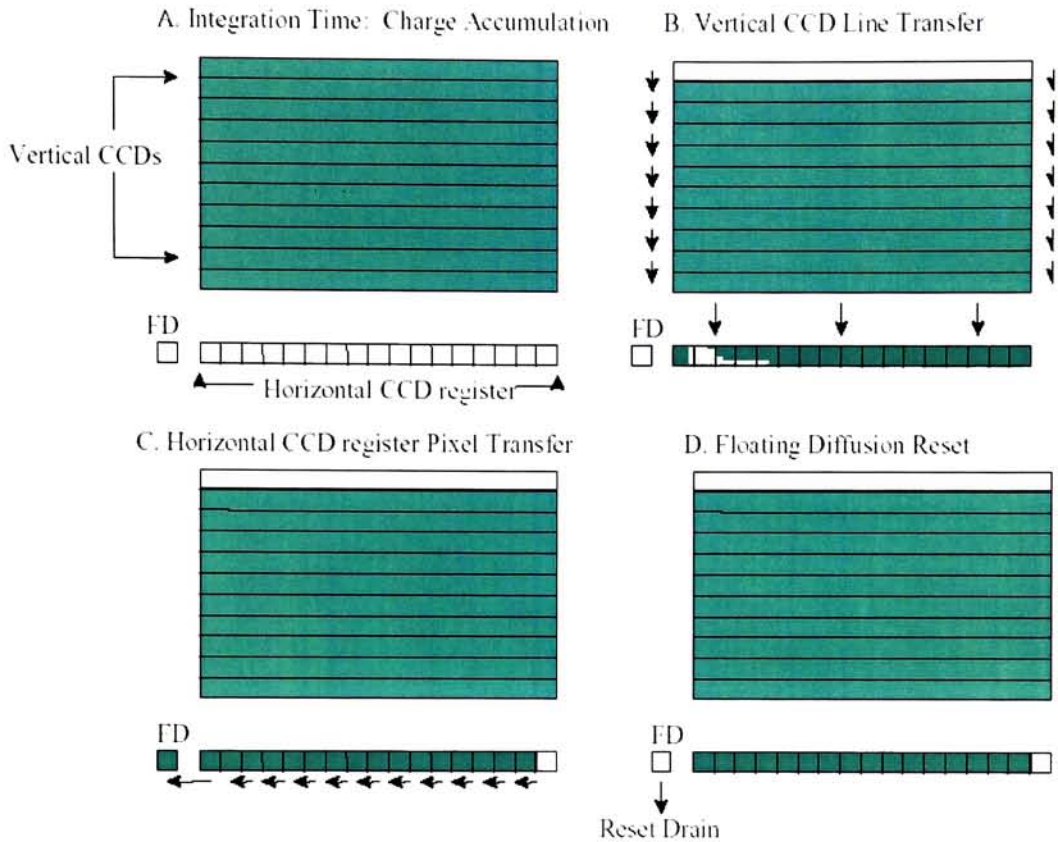

Figure 2-6: Full Frame CCD – Normal Data Clocking

When controlling exposure with a FF device, either a mechanical shutter mechanism or a synchronized strobe illumination source is required. This is due to the fact that, during the readout cycle of the FF CCD, the charge is shifted down the device. Exposing the CCD during this readout cycle can lead to image smearing.

A full frame CCD has 100% fill factor. This indicates that 100% of the pixels on the FF CCD are utilized in detecting photons.

FT CCDs builds off the same concepts as a FF CCD. One difference is that a FT CCD has its parallel register divided into two distinct areas as shown in Figure 2-7 [3]. Region A is the area where the image is focused/exposed, while Region B is the storage array.

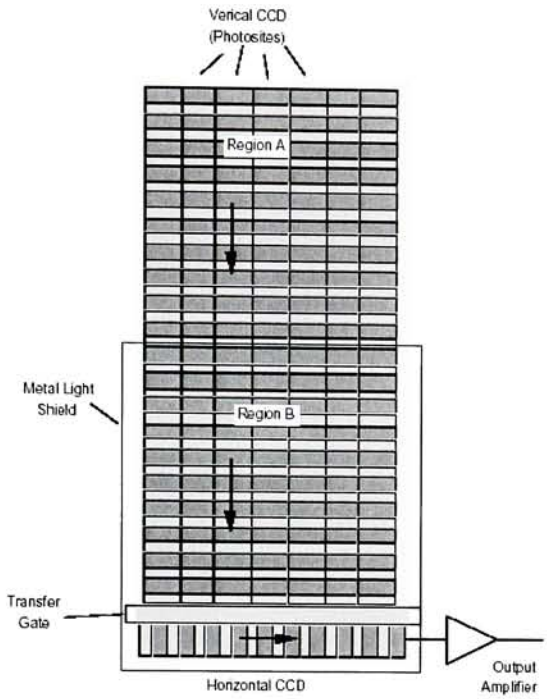

Figure 2-7: Frame Transfer CCD Sensor

After the image is exposed in region A, the image is quickly transferred to region B which eliminates the need for the mechanical shutter or synchronized strobe required by the FF CCD sensors.

FT CCD devices tend to be more expensive to manufacturer. They are twice the size of the full frame devices, thereby leading to a lower yield. Additionally, they have only a 50% fill-factor. FT CCDs are most commonly used in scientific imaging applications.

Interline CCDs utilize the same concept of a FT CCD, and address several of the FT shortcomings [2]. An interline CCD device shown in Figure 2-8, alternate photosensitive columns with adjacent columns of non-photosensitive storage regions. After exposure, the signal collected in every pixel is rapidly transferred, all at once, into the storage regions. Like FT devices, interline CCDs allow images to be read out, while the next image is captured, enabling a higher frame rate. Image smear during readout is significantly reduced in interline devices from that of FT CCDs [3].

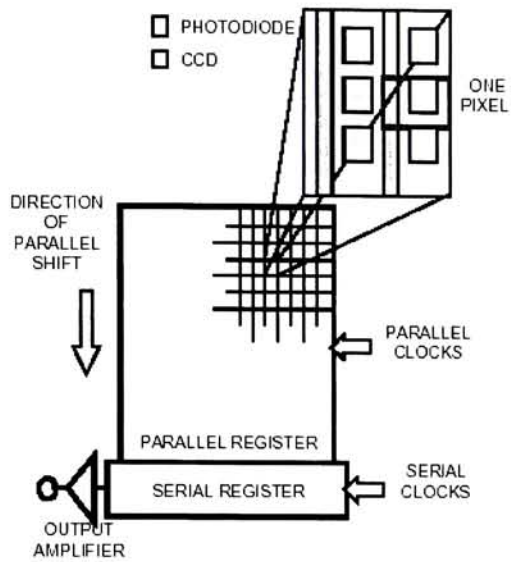

Figure 2-8: Interline CCD Sensor

Interline CCD devices are commonly found in lower-end digital cameras, video cameras, and broadcast cameras for motion capture [5]. Their short exposure times lend themselves to applications for imaging fast moving objects. The major disadvantages of interline CCDs are that their complexity leads to higher unit cost, and a lower sensitivity from the reduced photosensitive area.

2.2 CCD Specifications and Parameters

The following sections discuss CCD terms and definitions. The terms detailed here are ones that impact system performance [8], [10], [17].

2.2.1 CCD Terms & Definitions

1) Pixel Binning is the process of combining charge from adjacent photo sites on the CCD during readout. This process is performed prior to digitization on the CCD device by modification of the clocking sequence to the serial and parallel shift

registers. Pixel binning is sometimes referred to as a "super pixel". Further binning can take place in either H/W or S/W after digitization.

2) Signal-to-Noise Ratio (SNR) describes the relative magnitude of the signal compared to the uncertainty in that signal on a per pixel basis. SNR is sometimes referred to as dynamic range. SNR is range is a log ratio of full well depth to the readout noise in decibels at a pixel. SNR is very important in applications requiring precise light measurements. The SNR may be calculated using Equation 2-1.

$$S/N = 20 \log (\text{signal/noise})$$

Equation 2-1: Signal to Noise Equation

For a 12-bit system the theoretical maximum SNR is calculated using Equation 2-2.

$$SNR_{max} = 20 \log(\text{max signal/min noise}) = 20 \log (2^{12}/1) = 72\text{db}$$

Equation 2-2: Maximum SNR for a 12-bit System

3) System Gain is the number of electrons represented by each A/D count. In general, the lower the gain, the better. Adjusting the system gain is a compromise between the extremes of high digitization noise and the loss of well depth. The system gain may be calculated using Equation 2-3.

$$\text{System Gain} = \text{Electrons/Analog to Digital Units}$$

Equation 2-3: System Gain Equation

4) Charge transfer efficiency (CTE) is the fraction of electrons passed to the next position during the readout process. The ideal efficiency is 1.0, which indicates no electrons have been left behind.

5) Full-well capacity or saturation are terms used to describe the maximum number of electrons which can be stored by a pixel in a CCD. In order to support binning modes, this holding capacity is usually greater for the horizontal shift register and

output node as compared to the individual pixel photo sites. If the pixel is filled beyond full well (over exposed), a blooming image artifact is observed. Full-well capacity is specified in electrons or mV of output signal.

6) Blooming is the overexposure of the CCD, which causes too many free electrons to be produced in a pixel. This causes the pixel element to leak charge into less charged neighboring pixels. This artifact usually causes streaking from the bright scene content on the image being displayed.

7) Linearity is a measurement of how consistently the CCD responds to light over the range of exposure conditions. The average CCD signal output is measured over a range of integration times, up to full well depth. A straight line represents the ideal case for CCD linearity.

8) Quantum Efficiency (QE) is a term used to specify the percentage of photons incident on a CCD device that are detected, and always less than 100%. The QE is a function of the wavelength of the incident light. This same measurement is sometimes referred to as CCD sensitivity.

9) Fill Factor is a CCD metric that indicates what percentages of the total pixels are being utilized to detect photons.

10) Resolution in a scanner device is its pixel density. It is the number of active photo sites available to sample and reproduce an image.

11) Picture element (pixel) is an individual photo site that has the ability to collect a discrete amount of light (photons).

12) Integration time or exposure time are terms that are used interchangeably. These terms represent the amount of time the CCD is exposed to light. Depending on CCD format and device manufacturer, some include the readout time in this parameter.

13) Readout rate is the inverse of the serial conversion time. It is the time required to digitize a single pixel. Readout rates are usually given in pixels/second. In area CCD devices, it refers to the total time to read the entire CCD.

14) Frame rate is the inverse of the time needed for the CCD to acquire an image and then completely clock that image out. Frame rate is typically expressed in frames per second (fps).

15) Bit depth is the number of permutations or possible states which can be reproduced; therefore, having 12 bits results in 2^{12} possible states. That gives a light intensity range of values from 0 (black) to 4095 (maximum saturation).

16) Spectral responsivity of a CCD describes the sensitivity of a sensor as a function of input illumination wavelength.

2.2.2 CCD Noise Sources & Corrective Steps

The imaging qualities of CCD sensors are significantly impacted by the presence of noise in captured images [10]. SNR is the most important parameters impacting the performance of an imaging system. Due to this importance, additional discussion on noise sources and preventative measures will be described here.

The following is a brief list of sources of image noise in CCD devices, and corrective steps to minimize each noise source.

1) Readout noise is an error in reading the electronic signal and is largely due to noise that is injected by the output node/amplifier. This noise source is dependant on the internal design and construction of the CCD. Preventative steps the end user can take to limit this noise source is to utilize on sensor pixel-binning functions, if available [6].

2) Dark current or dark noise is due to the fact that in the absence of illumination, electrons are still thermally generated. It is a function of the CCD characteristics and temperature. Unlike readout noise, pixel binning does not impact the amount of dark

noise. A general estimation rule is that the dark noise doubles for each rise of 5 – 6 degrees C. Cooling a CCD, reducing exposure and readout times minimize the contribution of this noise source [6]. Depending on cost constraints of the imaging system, cooling may not be practical. Dark noise is given by Equation 2-4 [10].

$$\text{dark noise} = (\ (\text{dark current}) * (\text{integration time})\ )^{1/2}$$

Equation 2-4: Dark Noise Equation

3) Photon noise or shot noise is due to the uneven distribution of photons available for detection. A Poisson distribution function models the statistical nature of the uncertainty in the amount of photons collected per a given amount of time [7]. Photon noise is calculated using Equation 2-5 [10].

$$\text{Photon noise} = (\text{signal})^{1/2}$$

Equation 2-5: Photon Noise Equation

4) Pixel response non-uniformity noise is due to fact that the pixels of the CCD differ in sensitivity. This noise source is dependent on the internal design and construction of the CCD. Performing a flat field scan minimizes this noise source [7]. Flat fielding involves taking a correction exposure on a uniformly lit flat field. The raw image scan can then be adjusted by the amount determined in the flat field scan.

5) Quantization noise is errors in conversion of the signal from analog to digital (A/D). Greater bit depth on the A/D will reduce this noise source [6].

Very often, higher quality imaging systems utilize an additional noise preventative step in the analog clocking circuitry. A correlated double sampler circuit utilizes a signal processing technique that subtracts the CCD output signal black level from the video level. The correlated double sample circuit rejects common mode and power supply noise.

SNR increases are observed by increasing the signal level while reducing the noise sources. The following sections will discuss improving the SNR by increasing the signal level through on sensor pixel binning.

2.3 Multiple Resolution Applications with Pixel Binning

In this section, the Pixel Binning concept for data reduction will be further defined, and extended to both on and off sensor data reduction. Additionally, this section will discuss and briefly review recent applications and findings for data reductions from other journals and publications.

2.3.1 Pixel Binning Defined

Pixel binning is a clocking arrangement used to combine charge packets from neighboring pixels before sending them to the output amplifier. This reduces the spatial resolution of the CCD, but provides an improvement in the signal to noise ratio for low light conditions. The addition of many charge packets reduces the noise level by averaging out the pixel to pixel variations [8].

With binning enabled, the CCD clocking technique is modified. Consider the following example of 2 x 2 binning with a FF CCD sensor. Figure 2-9 may aid in the following description [18]. The FF CCD clocking will be modified so that two rows are shifted and summed in the horizontal shift register. At this point each pixel in the horizontal shift register would hold the summation of two pixels. Next, the horizontal register is clocked twice, allowing two of the horizontal pixels sums to be summed in the output node. Hence in the given example, the output amplifier digitizes the sum of electrons in the output node on every other horizontal shift clock pulse.

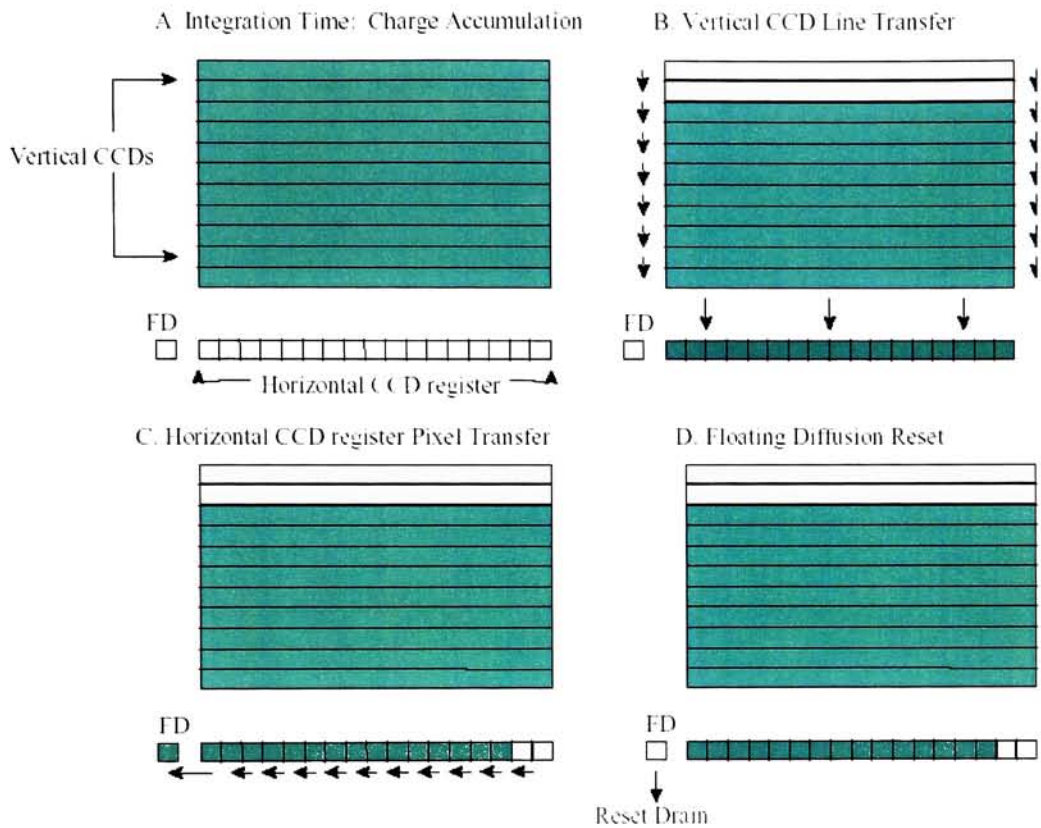

Figure 2-9: Full Frame Binning Example – 2x2 Mode

The resulting image from this example has the following characteristics: the amount of data is one quarter of the size of the original image, and the light sensitivity increased 4 times. Again, the price of this was a reduction in spatial resolution

Figure 2-10 shows the increase in pixel size based on 1x1, 2x2 and 4x4 binning mode of operation [12].

Figure 2-10: Pixel Size Differences for Several Binning Modes

Binning decreases readout time, increases sensitivity, and decreases image size. Pixel binning help to increases the frame rate for applications requiring timely updates.

2.3.2 Limitations to on CCD binning

Charge holding capacity of the horizontal shift register and output node extensively limits a CCD binning capabilities. Sometimes the CCD exposure time is reduced to avoid blooming in the horizontal shift register or output node structure.

2.3.3 Applications of Pixel Binning and Data Reduction

Welch's findings support that deletion of unwanted data close to the source is desirable for spacecraft imaging applications [13]. He indicates that it positively impacts a system by both minimizing processor time and more efficient utilization of communication channel bandwidth for space based applications. In addition to pixel binning, his findings support the following techniques: 1) thresholding of bad pixels, 2) removal of known corrupt pixels, and 3) avoid split event reconstruction.

French's (et al) findings indicate the advantages of compact versatile electronics for ground and space based astronomy and earth observation [15]. They designed a CCD controlling ASIC which substitutes for the general-purpose DSP or microprocessor solutions.

They also utilized pixel binning and windowing for minimization of data for their Space-borne CCD camera system.

Roberts' findings support innovative ways for inspection of wide-web industrial manufacturing applications [16]. He found that limiting data transmission to only detected defects in a region of interest reduced his H/W and data analysis S/W by a factor of 100. He utilized run length encoding to achieve a data reduction of 99% over conventional approaches.

Chapter 3
Pipelined Pixel Binning – H/W System Architecture

A highly pipelined H/W implementation is shown in Figure 3-1. This implementation supports off-sensor pixel binning, post digitization. At the highest level, it consists of an ASIC and 3 memory elements.

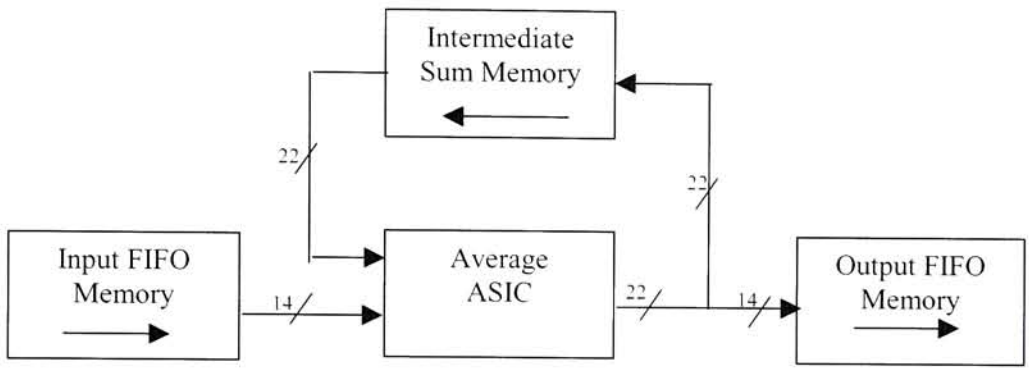

Figure 3-1: H/W System Architecture for Binning

In this architecture, the Input and Output memory devices isolate the speed of the averaging circuitry from that of the surrounding logic, allowing for asynchronous operation with upstream and downstream circuitry. The Averaging ASIC computes the two-dimensional average, as well as providing overall control of the entire data pipeline and memory elements.

Synchronous design techniques are utilized, such that all averaging circuitry is run from a single master clock. All events occur based on the rising edge of the system clock. A synchronous system reset signal, when asserted, resets all circuitry including the memory devices to an initial state. All signals shall maintain circuitry setup and hold time to avoid the meta-stability condition. The reset signal must also be asserted when changing averaging modes.

Synchronous First-In-First-Out (FIFO) memory devices best suit the data storage requirement for this implementation. Accessing data from these types of memory devices requires that the access control enable signal (read or write enable line) be asserted, coincident with the rising edge of the access clock signal (read clock or write clock). Multiple data access can be accomplished by asserting the access control enable signal for multiple clock cycles.

Synchronous memory devices allow for speed flexibility in adjacent circuitry. Upstream data may be written to the input FIFO memory at the rate at which it is clocked off the sensor and digitized. Theoretically speaking, the inputs FIFO write frequency must be slower than the read frequency (the rate at which the averaging circuitry processes data), otherwise throttle control techniques are required. Failing to implement data flow control measures, could result in the input FIFO over flow condition. In many systems, the data rate from the CCD is typically slower than the averaging circuitry processing data rate.

A similar condition exists with the output FIFO. Data is written to the output FIFO memory at the averaging circuitry processing data rate. Down stream logic must read the data from the output memory FIFO device to insure that the Output FIFO never overflows. Otherwise, throttle control techniques will be required to stall the Averaging ASIC pipeline.

The Averaging ASIC provides control signals necessary to perform the following memory functions: reading access from the Input FIFO Memory, read and write access to the Intermediate sum FIFO memory, and write access to the Output FIFO memory device.

3.1 H/W Implementation Differences for Full Frame and Tri-Linear Sensors

The implementation of off-sensor binning solutions is somewhat simplistic for full frame CCD devices. Having the image data organized in color planar form, the 2D averaging function involves summing adjacent pixel rows and columns. The red, green and blue color planes are each processed sequentially. This can be accomplished in a single time multiplexed arithmetic logical unit (ALU) and data path.

The pixel-interleaved organization often used with tri-linear CCDs present an interesting challenge for off CCD binning solutions. Unlike the previous approach, red, green, and blue pixels must be tracked and combined with only that of the correct color plane. All three planes of data must share the ALU during three concurrent averaging functions.

3.2 Data Path Requirements

Two control bits are necessary to be buffered with pixel data to encode if this pixel is a vertical or horizontal synchronization pixel. The vertical sync signal indicates that a pixel is the first pixel in a frame. The horizontal sync signal represents the first pixel in a line. The horizontal and vertical sync control signals will both be asserted for the first pixel of a frame. Horizontal and vertical synchronization signals must be maintained in the averaging logic, such that the output represents a two-dimensional averaged version of the input data.

A picture element or pixel is comprised of red, green and blue component colors. Data is arranged in a sequential color interleaved fashion. Hence, the sequential data stream will be that of repeating sequence of red, green, and blue component values for each consecutive pixel.

Pixel component information is represented using a 12-bit unsigned number; therefore, the allowable range of values is from 0 to 4095. A wider data bus is necessary for the pixel summation circuitry in order to store large intermediate sums. A maximum of 256, 12-bit pixel values may be summed for the 16 by 16 averaging mode. Intermediate summing circuitry must be 20 bits wide to support these larger sums. Additionally, two bits are required for storing the horizontal and vertical control signals.

3.3 Memory Storage Requirements

The limiting case, which determines the depth of the intermediate sum memory, is based on the condition where 2 by 2 averaging of a 2048 x 3072 input image is desired. The intermediate sum memory must store 1536 intermediate pixel pair sums under these

conditions. This represents half of the maximum input resolution width. As stated earlier, the width of intermediate FIFO must at least 22 bits wide (2-bits for control, plus 20-bits data). A standard FIFO device size is 2K x 18; therefore, two of these IC devices could be used to form the 22-bit intermediate storage buffer.

The input and output memory size requirements are based on the relative speed of the averaging circuitry in comparison to surrounding logic. The size or depths of these memory elements are chosen to avoid data overflow conditions; consequently, the Input Memory device size would be based on the write rate data rate versus the averaging circuitry reading data rate. The Output Memory device size would be based on the averaging circuitry write rate versus the Output Circuitry readout rate. Physically, a single FIFO device is sufficient since only 14 bits of data are stored (MSB to LSB: 1-bit vertical sync, 1-bit horizontal sync, and 12 bits data).

Under certain operating conditions, the input and output memory buffer circuitry may be replaced by a single register. Additionally, a register could replace the input memory buffer if the averaging circuitry can consume the data at a rate equaling or exceeding that of the data source. A register could replace the output memory buffer, if the output logic can consume the data at a rate equal to or exceeding that of which the averaging circuitry produces the data.

3.4 Averaging ASIC VHDL Model

The Averaging ASIC was written using Esclade Design Book tools. These Computer Aided Design tools provide several design entry methods including: graphical schematic, state machine bubble diagram, flow chart, Boolean truth table, and VHDL text. Esclade Design Book outputs synthesizeable VHDL or Verilog code. This code may then be synthesised to several ASIC manufactures silicon for multiple silicon device families/architectures.

The Averaging ASIC was written in a hierarchical fashion. Lower level components are hidden from the user, and will be described in more detail in the following sections. As shown in Figure 3-2, there are several inputs and outputs to the averaging function. A summary description of the Averaging ASIC I/O signals shown in Table 3-1.

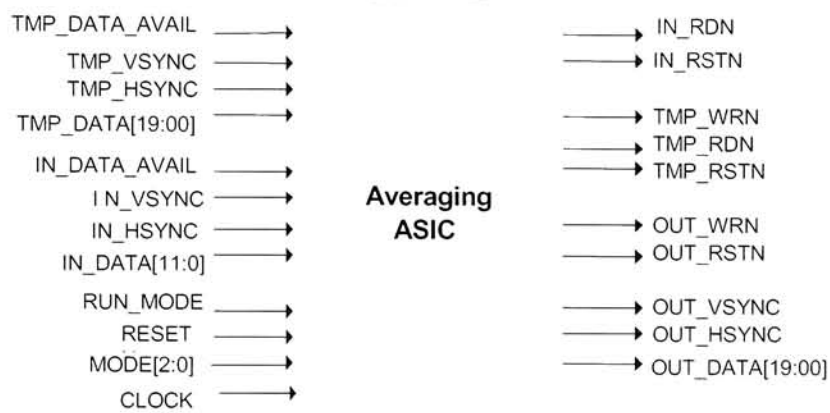

Figure 3-2: Averaging ASIC I/O Assignment

SIGNAL NAME	INPUT/OUTPUT	FUNCTIONAL DESCRIPTION
CLOCK	Input	System clock.
MODE[2:0]	Input	Encoded averaging mode
RESETN	Input	System resets signal which when asserted clears all registers. All state machines are set to an initial state.
RUN_MODE	Input	When asserted, run mode enables the Averaging ASIC to process data.
IN_DATA_AVAIL	Input	When asserted, indicates that data held in the input FIFO is available for processing.
IN_DATA[11:0]	Input	12-bit color interleaved pixel input data bus.
IN_HSYNC	Input	Input horizontal synchronization signal that

SIGNAL NAME	INPUT/OUTPUT	FUNCTIONAL DESCRIPTION
		indicates the first pixel of a line.
IN_VSYNC	Input	Input vertical synchronization signal that indicates the first pixel of a frame.
TMP_DATA_AVAIL	Input	When asserted, indicates that intermediate sum pixel data held in the temporary FIFO is available.
TMP_DATA[19:0]	Input	20-bit color interleaved pixel intermediate sum data bus.
TMP_HSYNC	Input	Temporary sum horizontal synchronization signal that indicates the first pixel of a line.
TMP_VSYNC	Input	Temporary sum vertical synchronization signal that indicates the first pixel of a line.
IN_RDN	Output	Input FIFO read enable signal is asserted to read a pixel from the input FIFO on the next rising edge of the clock.
IN_RSTN	Output	Input FIFO reset signal is asserted during a system reset to clear the contents of the input FIFO.
TMP_WRN	Output	Temporary FIFO write-enable signal is asserted to write an intermediate sum pixel to the temporary FIFO on the next rising edge of the clock.
TMP_RDN	Output	Temporary FIFO read-enable signal is asserted to read an intermediate sum pixel from the temporary FIFO on the next rising edge of the clock.
TMP_RSTN	Output	Temporary FIFO reset signal is asserted during a system reset to clear the contents of the temporary FIFO.
OUT_HSYNC	Output	Output horizontal synchronization signal that indicates the first pixel of a line for intermediate sums or output results.
OUT_VSYNC	Output	Output vertical synchronization signal that indicates the first pixel of a frame for intermediate sums or output results.
OUT_DATA[19:00]	Output	20-bit color interleaved pixel data bus for intermediate sum or output results.

Table 3-1: Averaging ASIC I/O Description

A signal naming convention has been adopted such that any active high signal name will ends in any letter other than "N". All active low signals must end in the letter "N". Active low signals are usually reserved for critical control signals, where greater noise immunity is desired. Active low signals when de-asserted are in a high voltage state and are driven to a low voltage level when asserted.

The Averaging ASIC design and adjacent memory devices operate from a single system clock (CLOCK). The following is a brief description of its input signals. The reset signal (RESETN), when asserted, clears all logic registers and state machines to an initial value. Mode[2:0] is the encoded averaging mode. The Mode[2:0] input pins control the amount or area of 2D averaging. As shown in Table 3-2 below, the output resolution is both a function of input resolution and the Mode[2:0] setting.

Input Resolution (H x W pixels)	M[2:0] = 000 No Averaging	M[2:0] = 001 2 x 2 Area	M[2:0] = 010 4 x 4 Area	M[2:0] = 011 8 x 8 Area	M[2:0] = 1xx 16 x 16 Area
1024 x 1536	1024 x 1536	512 x 768	256 x 384	128 x 192	64 x 96
2048 x 3072	2048 x 3072	1024 x 1536	512 x 768	256 x 384	128 x 192

Table 3-2: Output Resolution as a Function of Input Resolution and Averaging Mode [2:0]

The IN_VSYNC, IN_HSYNC, and IN_DATA[11:00] signals are the sequential color interleaved pixel data stream to be averaged. The IN_DATA_AVAIL signal is asserted indicating that the input memory device has valid data to be averaged. The TMP_VSYNC, TMP_HSYNC, and TMP_DATA[19:00] signals are the sequential color interleaved intermediate sum pixel data stream. The TMP_DATA_AVAIL signal is asserted indicating that the intermediate (or temporary) memory device has a valid intermediate sum to be added to the incoming pixel data stream. Before the Averaging ASIC will process any data, the run mode input signal must be asserted.

The following is a brief description of Averaging ASIC output signals. The OUT_VSYNC, OUT_HSYNC, and OUT_DATA[19:00] signals are the sequential color interleaved pixel data stream to be stored in the output or intermediate holding FIFO. The intermediate buffer is written to when additional pixels need to be summed. The output buffer is written to when an averaged pixel is generated. OUT_WRN and OUT_RSTN signals are asserted for writing and resetting the output memory buffer respectively. TMP_RDN, TMP_WRN and TMP_RSTN signals are asserted for reading, writing, and resetting the intermediate (or temporary) memory buffer respectively. IN_RDN, and IN_RSTN signals are asserted for reading and resetting the input memory buffer respectively.

The Averaging ASIC may be broken down into two primary functional areas, the data pipeline and averaging control logic, as shown in Figure 3-3. The data pipeline path provides the routing and circulation of averaging data. The averaging control circuitry generates internal ASIC control signals, as well as controlling access to the external memory devices.

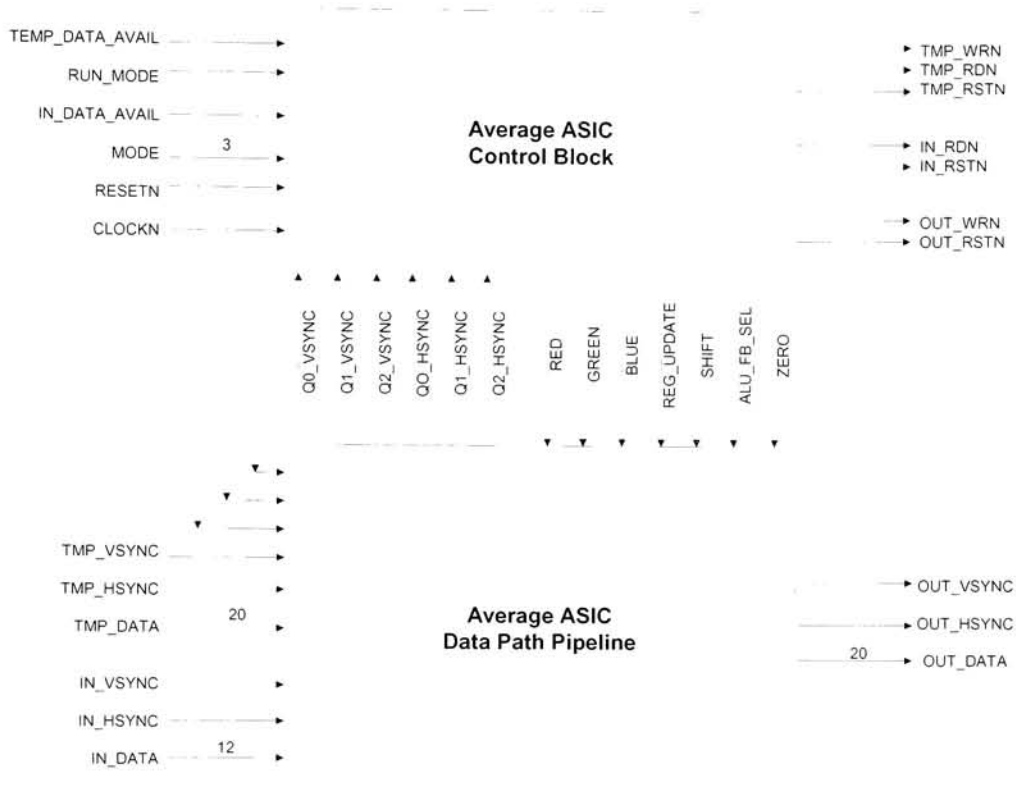

Figure 3-3: Average ASIC Block Diagram

3.4.1 Averaging ALU and Data Pipeline

The Averaging ALU and Data Pipeline are controlled entirely by the availability of data for processing in the input FIFO, and the Run_Mode input asserted. All registers are cleared to zero on the reset condition. The pipeline registers are updated when data presence is detected in the input FIFO, and it is being read out. Whenever the input FIFO is in the empty condition, the pipeline stalls and all registers maintain their current value. The pipeline registers retain their current value until additional data is read from the input FIFO. Each read cycle from the input FIFO presents a vertical sync bit, a horizontal sync bit, and a 12-bit pixel value.

Shown in Figure 3-4 below, the Averaging ALU and Data Pipeline is made up of the following components: Reg14, ALU, Reg22, Accum_Rmux, and Reg22_Dshifter. Each of

these components will be described in the following sections. The register update (REG_UPDATE) internal control signal must be asserted for the movement of any data in the pipeline. Otherwise, all pipeline registers maintain the currently held data value.

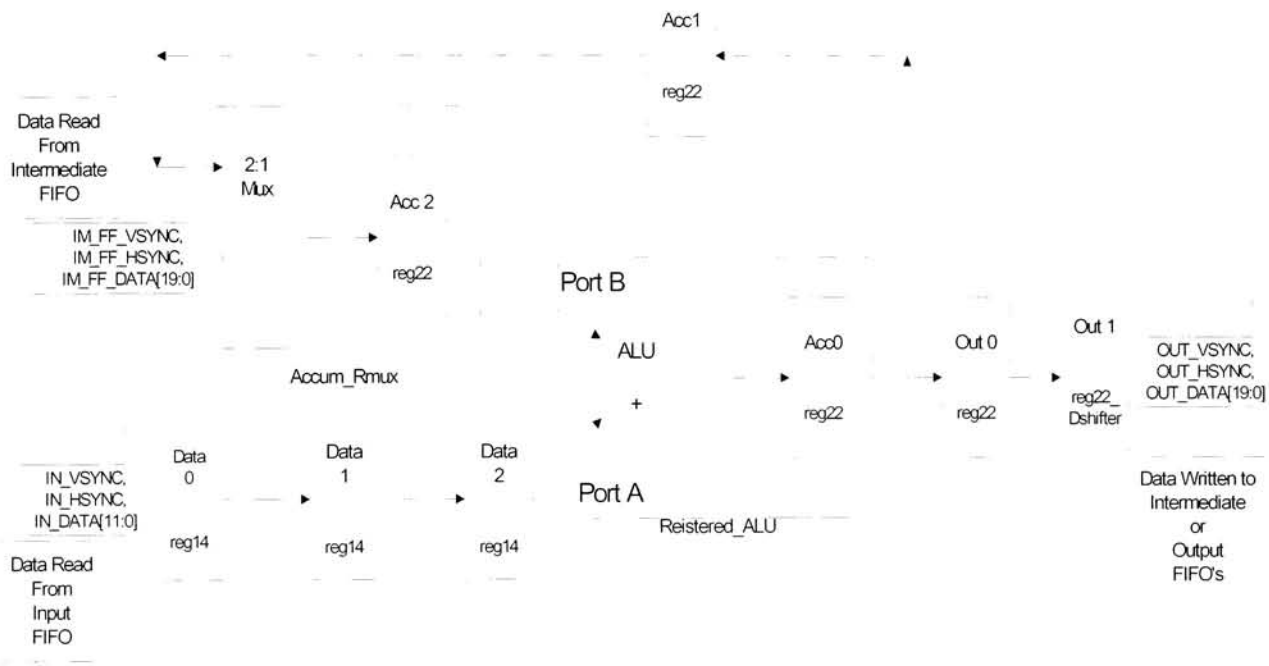

Figure 3-4: Averaging ASIC - Internal Data Path and ALU

3.4.1.1 Reg14

The Reg14 component shown in Figure 3-5 is a synchronous 14-bit register/latch that is clearable. The contents of this register are updated on the rising edge of the CLK signal when the REG_UPDATE input is asserted. When the synchronous reset input signal is asserted (RESETN), the contents of Reg14 are cleared to zero.

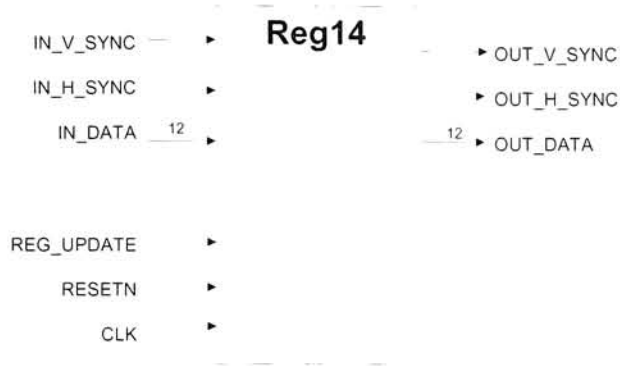

Figure 3-5: Reg14 Block Diagram

3.4.1.2 Reg22

The component shown in Figure 3-6 is a synchronous 22-bit register/latch, which is clearable. The contents of this register are updated on the rising edge of the CLK signal when the REG_UPDATE input is asserted. When the synchronous reset input signal is asserted (RESETN), the contents of Reg22 are cleared to zero.

Figure 3-6: Reg22 Block Diagram

3.4.1.3 Accum Rmux

The loop back accumulate registered mux component shown in Figure 3-7 is a synchronous 22-bit registered mux that is clearable. The contents of the mux registers are

updated on the rising edge of the clock signal when the clock enable (REG_UPDATE) input is asserted. Asserting the synchronous reset or zero input signal, clears the contents of the registered mux to zero.

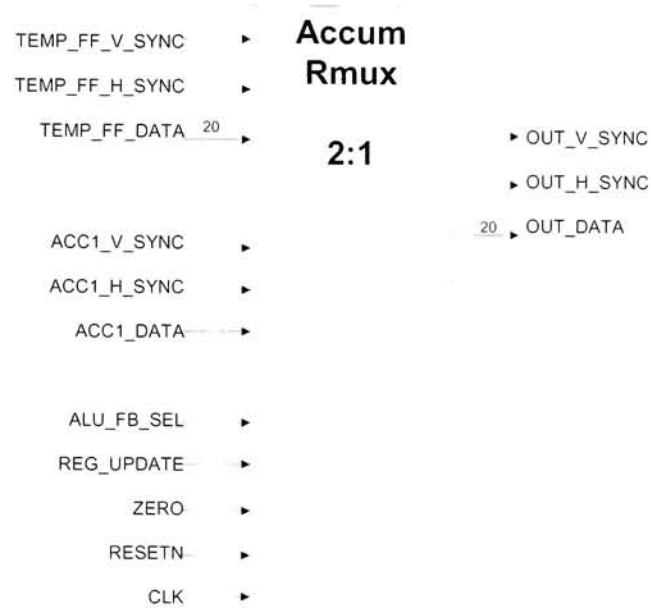

Figure 3-7: Loopback/Accumulate 2:1 Registered Mux

The state of ALU feedback select signal controls the mux output. When ALU_FB_SEL is asserted, the contents of the accumulate one register are sent to the mux output. When ALU_FB_SEL is cleared, the contents of the Temp FIFO are sent to the mux output.

3.4.1.4 ALU20

The 20 bit ALU component shown in Figure 3-8 implements a 20 bit registered adder function which is clearable. The output of the ALU is updated on the rising edge of the clock signal when the REG_UPDATE input is asserted. When the REG_UPDATE input is cleared, the ALU holds its current value.

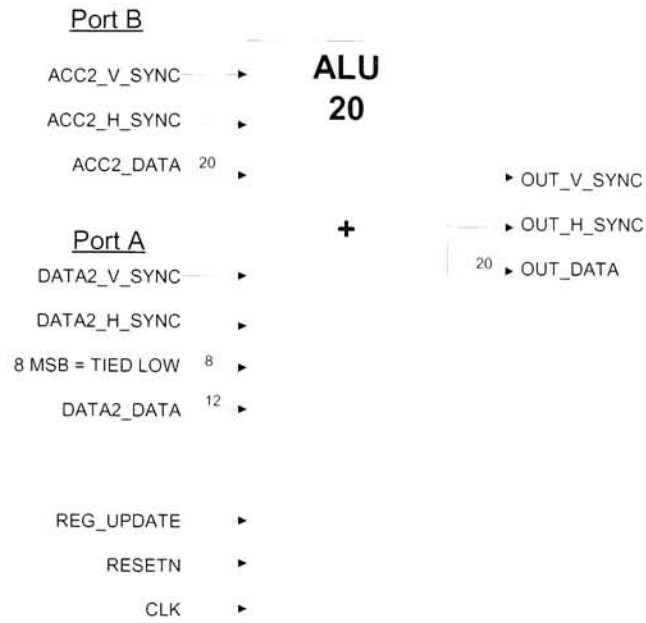

Figure 3-8: 20 Bit ALU Block Diagram

Port A is 12-bit input data along with the 8 MSB's tied to zero. Port B represents the 20-bit intermediate sum. This intermediate sum is either data looped back internally from the ALU output, or an intermediate sum value read from the intermediate sum FIFO.

The horizontal and vertical control signals at the output are represented as the "logical or" of the input.

3.4.1.5 Reg22_dshifter

The 22-bit register/data shifter component shown in Figure 3-9 is a synchronous 22-bit register/latch, which is clearable. The contents of this register/shifter are updated on the rising edge of the CLK signal when the REG_UPDATE input is asserted. When the synchronous reset input signal is asserted (RESETN), the contents of Reg22 Shifter are cleared to zero.

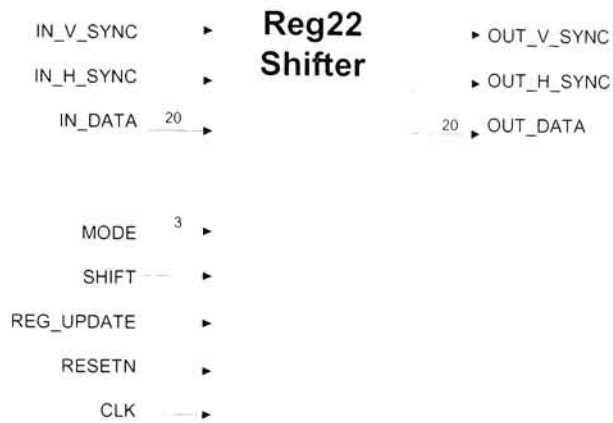

Figure 3-9: Reg22 Shifter Block Diagram

The shifting function allows for the following division based on mode of operation. The division function is enabled when the shift input is asserted. The amount of shifting is based on the 3–bit mode input and as specified in Table 3-3.

MODE[2:0]	Divided by	# of bits that the data bus shifted to the right by
000	1	None
001	4	2
010	16	4
011	64	6
1XX	256	8

Table 3-3: Reg22_dshifter Division based on Binning Mode

The horizontal and vertical sync control signals are passed through the Reg22_dshifter unmodified, regardless of the mode of operation.

3.4.2 FIFO Memory Interface Logic

The following sections detail the interface logic for the input, intermediate sum, and output FIFO's.

3.4.2.1 Input FIFO Interface Logic

The Averaging ASIC controls reading from the input FIFO and is shown in Figure 3-10. The format of the data stored in the input FIFO is vertical sync bit, horizontal sync bit, and 12 bits of image data. When the reset signal is asserted coincident with the rising edge of the clock, the input FIFO reset signal is asserted, and the FIFO contents are cleared.

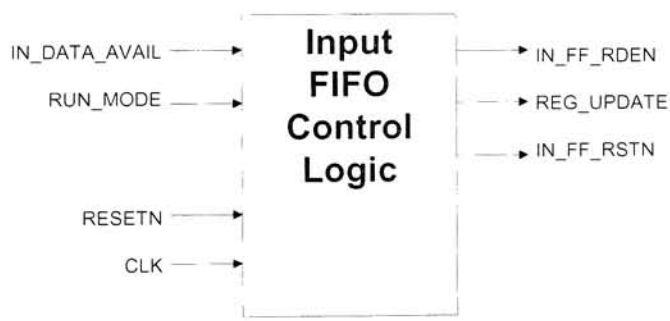

Figure 3-10: Input FIFO Interface Control Logic Block Diagram

The Averaging ASIC continuously monitors the state of the input FIFO's empty flag. When the input FIFO flag changes from the empty to not empty state and the run mode input signal is asserted, the read logic asserts the input FIFO read signal on the next clock edge. The input FIFO read signal will remain asserted until all data within the input FIFO has been processed.

When data is read from the input FIFO into the ASIC, the register update signal is asserted, and all internal registers in the data pipeline update their data values. The entire pipeline is controlled by the rate at which data is read from the input FIFO.

3.4.2.2 Intermediate Sum FIFO Interface Logic

The Averaging ASIC controls write and read accesses to the intermediate sum FIFO and are shown in Figure 3-11. The format of the data stored in the intermediate FIFO is vertical sync bit, horizontal sync bit, and 20 bits of image data. This FIFO stores the intermediate sum of several pixels from one or many lines. When the reset signal is asserted coincident with the rising edge of the clock, the intermediate sum FIFO reset signal is asserted, and the FIFO contents are cleared.

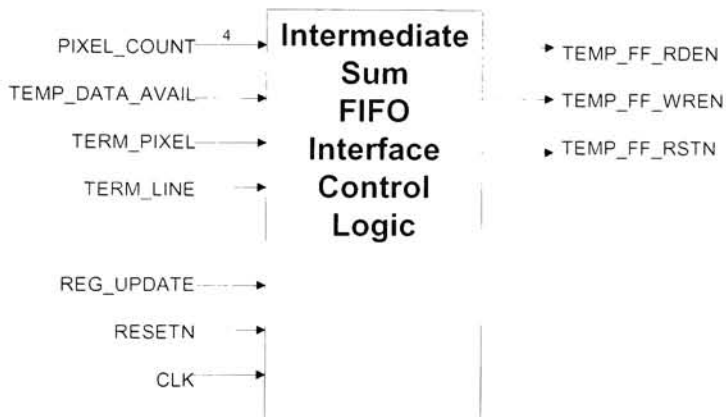

Figure 3-11: Intermediate Sum FIFO Interface Control Logic Block Diagram

The Averaging ASIC writes data to the intermediate sum FIFO when the pixel counter has expired, but the line counter has not. This condition designates that additional lines of pixels need to summed with the current intermediate sum value. Hence, a single value stored in the intermediate sum FIFO will be the sum of multiple pixels. The intermediate sum is

held in this FIFO until needed at a later time (for summation with neighboring pixels on the next line or lines).

The Averaging ASIC reads data from the intermediate sum FIFO when the following conditions are present: register update asserted, intermediate FIFO holds valid data, and the pixel counter has expired. This condition designates that an intermediate sum (or pixel value) is required to be added to the pixel currently being read from the input FIFO.

The utilization of the intermediate sum FIFO is dependent on the mode of operation. This FIFO is only utilized during binning operations. There is no need for temporary data buffering during Mode 000.

During binning modes, the write and read access rate to the intermediate FIFO are variable depending on mode of operation. The greatest access to this FIFO is observed during mode 001 where this FIFO is accessed for every other pixel being processed. During mode 1XX, the intermediate FIFO is be accessed once for every 16 pixel being processed. Table 3-4 below indicates the maximum contents of the intermediate sum FIFO based on mode of operation.

MODE[2:0]	Averaging of pixels	Maximum Contents of Intermediate Sum FIFO
000	None	Empty
001	2 x 2	½ input line resolution
010	4 x 4	¼ input line resolution
011	8 x 8	1/8 input line resolution
1XX	16 x 16	1/16 input line resolution

Table 3-4: Maximum Intermediate FIFO Storage Requirements

3.4.2.3 Output FIFO Interface Logic

The Averaging ASIC controls writing to the output FIFO and is shown in Figure 3-12. The format of the data stored in the output FIFO is vertical sync bit, horizontal sync bit, and 12 bits of image data. When the synchronous reset input signal is asserted, the output FIFO reset signal is asserted, and the FIFO contents are cleared.

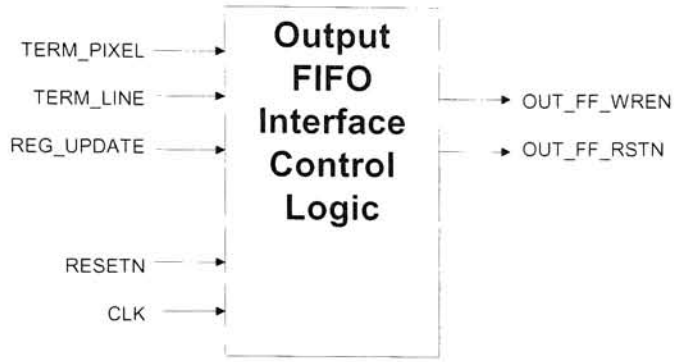

Figure 3-12: Output FIFO Interface Control Logic Block Diagram

The Averaging ASIC writes to the output FIFO when all pixels for the current mode have been summed and divided by the data shifter function. The output FIFO write enable signal is asserted when the following condition is detected: register pipeline update asserted, pixel counter at terminal count, and line counter at terminal count.

The frequency of writes to the output FIFO is variable and depends on the mode of operation. Most frequent access to the output FIFO are observed during mode 000 where this FIFO is written to for every pixel being processed. Least frequent accesses to the output FIFO is observed during mode 1XX, where this FIFO is written to once for every 256 pixels processed.

3.4.3 Data Path Control Logic

Several state machine controllers are used to properly direct the flow of data within the Averaging ASIC. The state machines utilized by the Averaging ASIC are: the wake-up state machine, color code state machine, pixel counter state machine, and line counter state machine. Each of these controlling state machines will be detailed in the following sections.

3.4.3.1 Wake-up State Machine

The wake-up state machine in Figure 3-13 controls zeroing the ALU Port B data bus during start up and data pass through modes. Figure 3-14 is a bubble diagram for this state machine. After the reset pulse, this state machine enters the S0 state, asserting the ZERO

output signal. It exits from S0 to S1 states, when the first vertical sync is detected while the pipeline update signal asserted. While in the S1 state, the zero output is asserted while in "no averaging" mode. The only way to exit the S1 state is by the reset pulse.

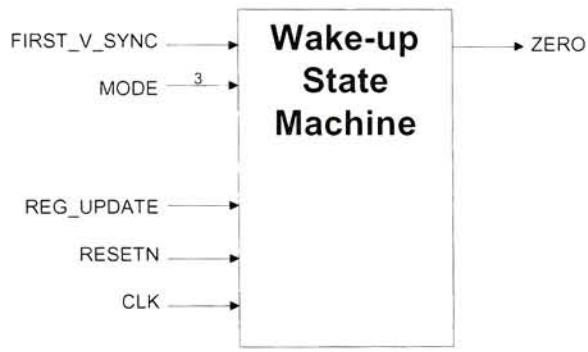

Figure 3-13: Wake-up State Machine Block Diagram

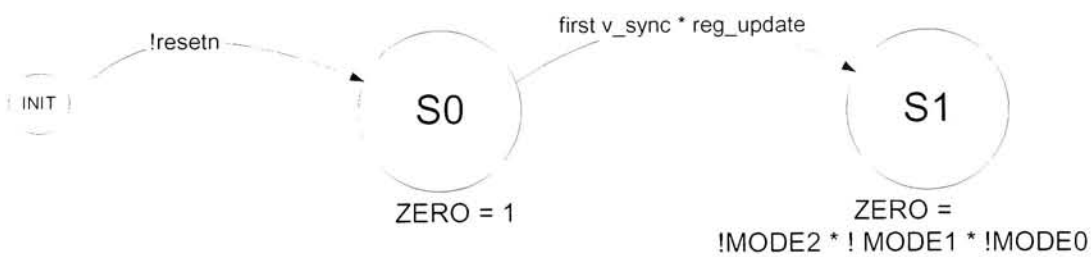

Figure 3-14: Wake-up State Machine Bubble Diagram

3.4.3.2 Color Code State Machine

The color code state machine in Figure 3-15 keeps track of which color plane the data actively being read from the input FIFO belongs to: red, green, or blue. Figure 3-16 is a bubble diagram for this state machine. Asserting the synchronous reset input causes this state machine to go to the "no color" or S0 state.

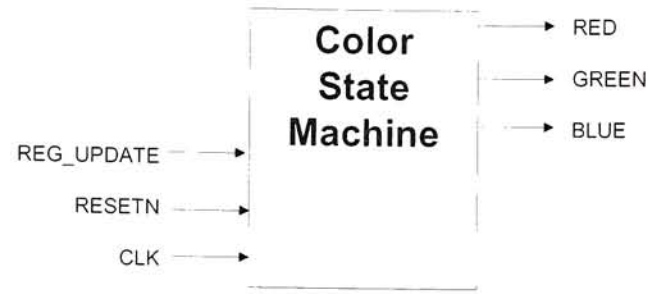

Figure 3-15: Color State Machine Block Diagram

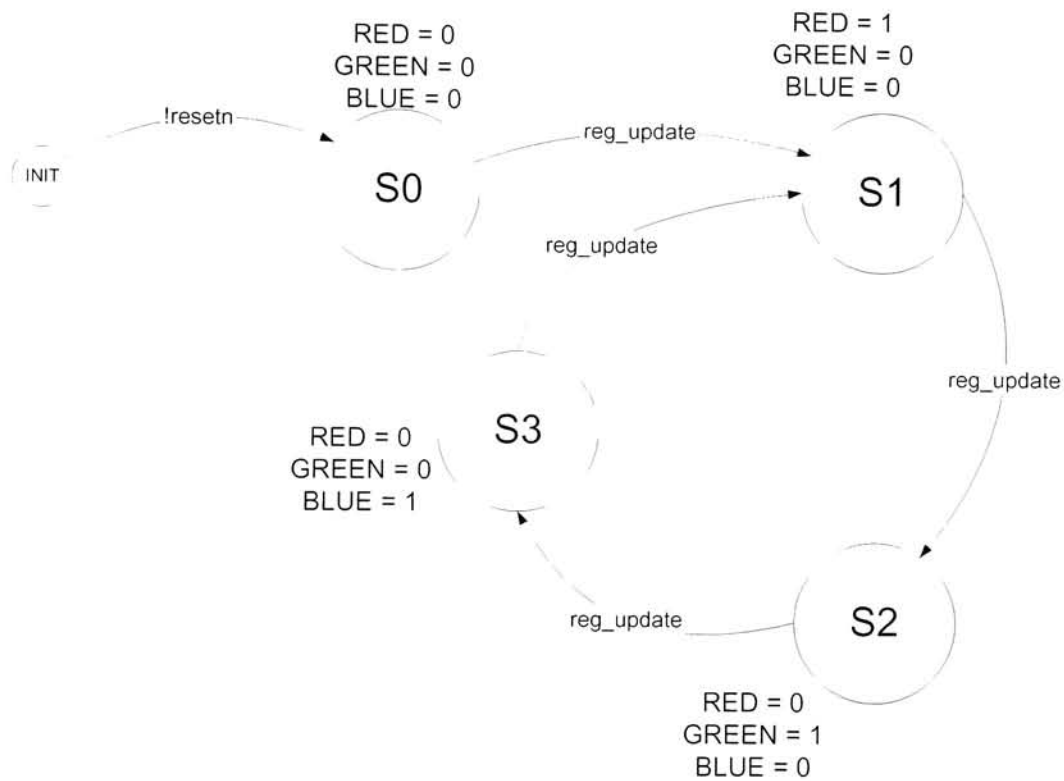

Figure 3-16: Color State Machine Bubble Diagram

The processing of the first and subsequent pixels is detected by the pipeline register update signal being asserted. Each assertion of this signal causes this "one hot" state machine to advance to the next state. First red, then green, then blue, then red… and so on. When the register pipeline is not being updated, this state machine stays in its current state.

3.4.3.3 Pixel Counter State Machine

Counting functions of the pixel counter state machine and that of the line counter state machine are very similar. The pixel counter state machine counts the adjacent pixels processed within a row. The line counter state machine counts the number of adjacent lines of pixels that have been processed.

The differences between these two functions were minimal. These differences included: differing update colors, and the requirement for the presence of a horizontal sync. These difference functions were identified early in the design phase and a single universal state machine constructed. This enabled two instances of the same state machine design to be utilized. Figure 3-17 is a block diagram of this universal state machine. Figure 3-18 is a bubble diagram for this state machine. Asserting the synchronous reset input signal causes this state machine to go to the "count0" state.

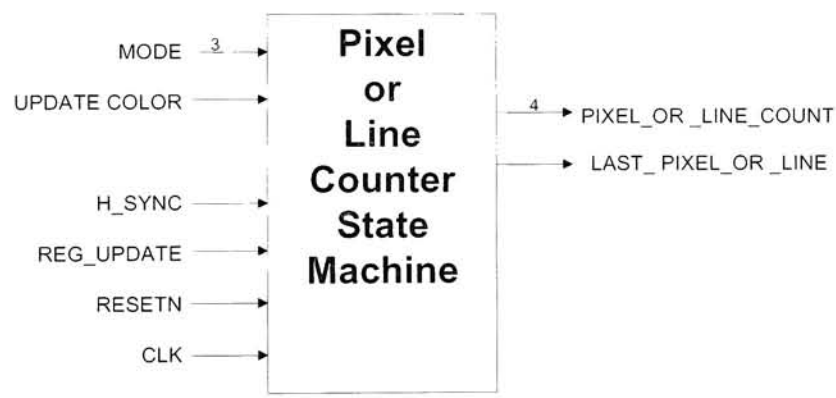

Figure 3-17: Pixel or Line Counter State Machine Block Diagram

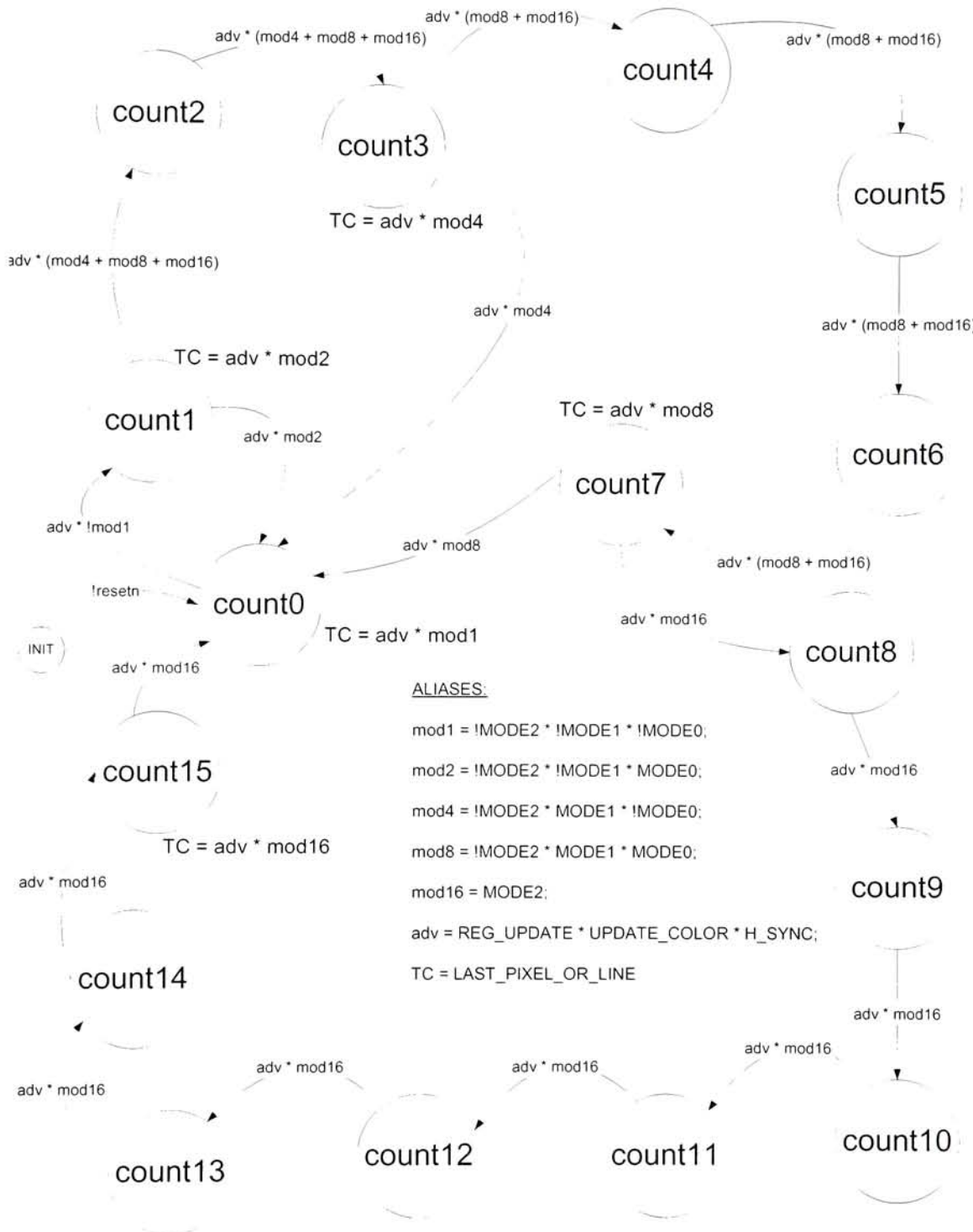

Figure 3-18: Pixel or Line Counter State Machine Bubble Diagram

The pixel counter state machine keeps track of the number of adjacent pixels within a line that have been summed together. This is dependent on the mode of operation. Table 3-5 below indicates the number of adjacent pixels summed together within the same line based on mode of operation.

MODE[2:0]	Number of pixels within the same line summed	Number of lines summed	Total Number of Pixels Summed Together
000 (no averaging)	1	1	1
001 (2 x 2)	2	2	4
010 (4 x 4)	4	4	16
011 (8 x 8)	8	8	64
1XX (16 x 16)	16	16	256

Table 3-5: Number of Adjacent Pixel and Lines Summed Based on Averaging Mode

The pixel counter state machine is updated when the blue plane signal is asserted. This signal indicates that another complete pixel triplet (red, green and blue) has been processed. The horizontal sync signal input to this state machine is not required and is tied to the asserted state for correct functionality. When the pixel counter is at its terminal count and the line count is not terminal, the current pixel summation is stored in the intermediate FIFO. If the pixel count state machine is not at a terminal count for the mode of operation, the intermediate sum data is looped back through the internal accumulator multiplexor.

3.4.3.4 Line Counter State Machine

The line counter state machine keeps track of the number of lines that have been summed together. The line counter state machine block diagram is shown in Figure 3-17 while the bubble diagram is shown in Figure 3-18. The number of lines summed together is variable

and depends on the mode of operation. Table 3-5 indicates the number of lines summed together and the total number of pixels summed together based on mode of operation.

The line counter state machine is updated when the red plane signal is asserted and the horizontal sync signal being asserted. This signal indicates that another complete line of data has been processed. When both the line and pixel counter state machines are at the terminal count, the current pixel summation value is stored in the intermediate FIFO.

Chapter 4
S/W Binning System Architecture

This chapter details the design of the S/W data averaging architecture. This source code was developed on a PC running the Windows 2000 operating system. These routines were developed using Microsoft Visual C/C++ development and debug environment.

4.1 Full Frame & Tri-Linear Solutions Identical

The S/W solutions for both FF and tri-linear CCD devices are identical. In the FF approach, the data is already presented to memory in color planar form. In a tri-linear CCD scanning implementation, usually three memory banks are used to convert the image data format from pixel interleaved to color planar formats. In both cases, a FIFO memory element holds the color planar data for an entire frame. A direct memory access cycle would expeditiously transfer this data from the FIFO memory to system memory, usually over the PCI bus.

The only difference in both approaches is that in the FF scanning approach, a single physical FIFO memory device is time multiplexed. In the tri-linear scanning approach, all three FIFO's are written to simultaneously. The read out cycle would "ping-pong" between the three memory FIFO devices, reading out each color plane.

4.2 S/W System Architecture

The S/W architecture chosen is displayed in Figure 4-1. In this figure, hollow arrows indicate user provided input or program generated output to a file. Thin arrows lines indicate passed data structures. The S/W is broken down into the following primary functions: get_inres(), init_input(), get_outres(), paxelize(), and store_output().

A main() function provides calls to the primary supporting functions.

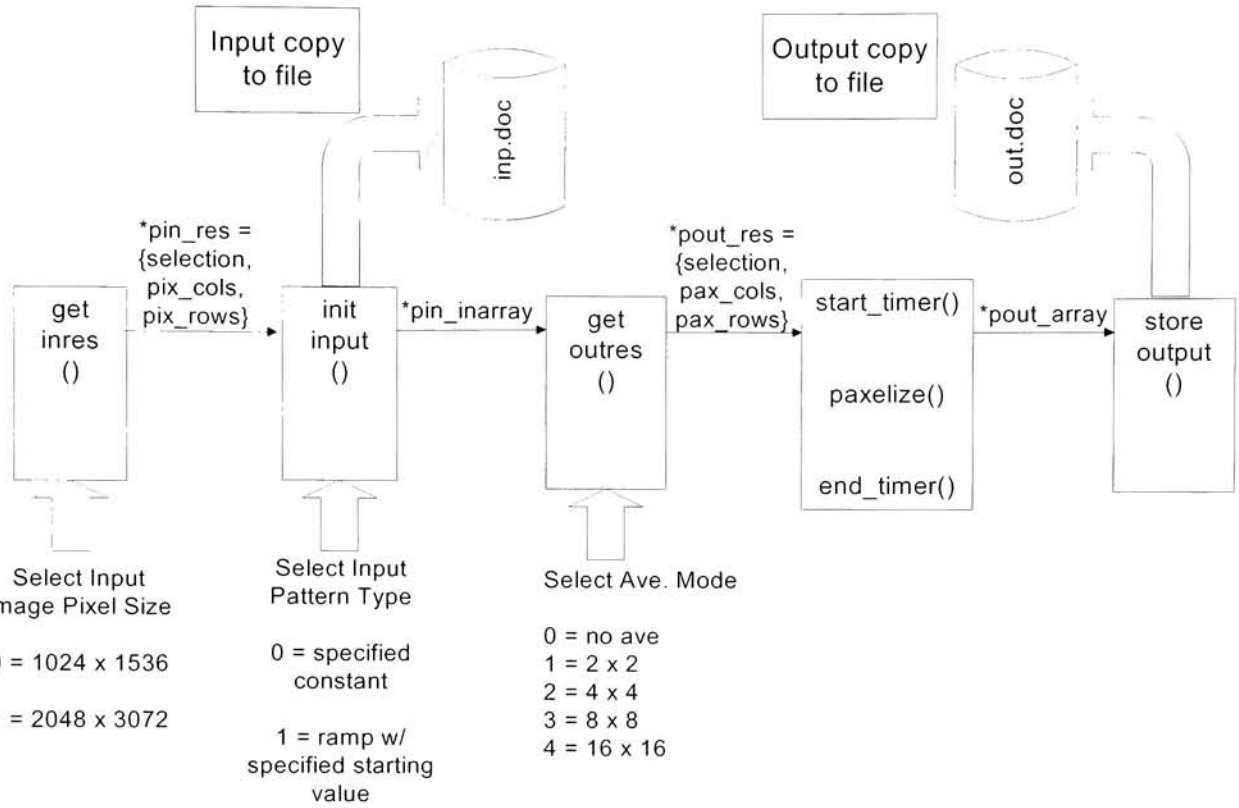

Figure 4-1: S/W Architecture

The get_inres() function prompts the user for the input resolution. Zero is selected for an input image size of 1024 x 1536 pixels, while one is selected for an input resolution size of 2048 x 3072 pixels. An input memory array is allocated according to the selected input resolution.

A RESOLUTION data structure is utilized to hold the input parameters. There are three integer members of this structure: selection – defines input resolution; pix_cols - defines the input column pixel dimension; and pix_rows defines the input row pixel dimension. A variable named "inresolution" is utilized to store these parameters.

The init_input() function prompts the user for what type of input data is to be generated. The user may specify the decimal value for either a flat field constant, or a ramping field.

Twelve bit numbers are utilized, therefore the acceptable range of input values are from 0 to 4095. When an incrementing ramp is specified, data is generated from the user specified initial value up to 4095. The remainder of the input array is initialized using a counter from 0 to 4095 (Modulo 4096). A copy of the input data array is written to a file named "inp.doc", for off line analysis and debug. The init_input() function returns a pointer to the input array.

The get_outres() function uses the input averaging array size parameter along with a user specified output averaging mode to generate an averaged output of the input array. Table 4-1 below are valid output averaging selections:

AVERAGE MODE SELECTION	# OF INPUT PIXELS AVERAGED FOR ONE OUTPUT PAXEL	COMMENT
0	none averaged (1:1)	no averaging (output size = input size)
1	2 x 2 averaged	output size = Ave (input size/4)
2	4 x 4 averaged	output size = Ave (input size/16)
3	8 x 8 averaged	output size = Ave (inputsize/64)
4	16 x 16 averaged	output size = Ave (input size/256)

Table 4-1: S/W Averaging Mode Selections

The get_outres utilizes a lookup table to define the output memory requirements. A RESOLUTION data structure is utilized to hold the output parameters, and is identical to that of the input. There are three integer members of this structure: output averaging selection - (0,1,2,3,4) defines output averaging mode; pax_cols - defines the output column dimension in pixels; and pax_rows defines the output row dimension in pixels. A variable named "outresolution" is utilized to store these parameters. Table 4-2 indicates the output resolution as a function of averaging mode for a 2048 x 3072 input image. This table also indicates the

reduction in the image sizes as a result to the averaging (input images size minus output image size).

INPUT RESOLUTION	AVERAGING MODE	OUTPUT RESOLUTION	IMAGE SIZE REDUCTION IN MB
2048 x 3072 (37.8 MB)	0	2048 x 3072 (37.8 MB)	0
	1	1024 x 1536 (9.4 MB)	28.4
	2	512 x 768 (2.4 MB)	35.4
	3	256 x 384 (0.6MB)	37.2
	4	128 x 192 (0.15 MB)	37.65

Table 4-2: Output Res. Based on Averaging Mode and Input Res. = 2048 x 3072

Table 4-3 indicates the output resolution as a function of averaging mode for a 2048 x 3072 input image. The number in parenthesis indicates the image size in mega bytes. This table also indicates the reduction in the image sizes as a result to the averaging (input images size minus output image size).

INPUT RESOLUTION	AVERAGING MODE	OUTPUT RESOLUTION	IMAGE SIZE REDUCTION IN MB
1024 x 1536 (9.4 MB)	0	1024 x 1536 (9.4 MB)	0
	1	512 x 768 (2.4 MB)	7.0
	2	256 x 384 (0.6MB)	8.8
	3	128 x 192 (0.15 MB)	9.25
	4	64 x 96 (0.037 MB)	9.0

Table 4-3: Output Res. Based on Averaging Mode and Input Res. = 1024 x 1536

The paxelize() function is the heart of the S/W algorithm. This function sequences through the input array and generates the averaged output array. The paxelize() function is passed in three pointer references. The first pointer points to the beginning of the input array data. The second pointer points to the inresolution data structure. The third pointer points to the outresolution data structure. The paxelize() function returns a pointer to the averaged output array.

System timing calls are utilized to measure S/W execution time of the paxelize() function. The first call to clock() initiates the measuring of time, effectively starting a timer. Calling clock a second time (after paxelize() function completes execution) provides another time measurement, effectively ending the timer. The difference of these two time measurements represents the execution time of the paxelize() function.

The paxelized() function sequences through a group of input pixels, as specified in the averaging mode, to generate an output paxel element. An output paxel element represents the two-dimensional averaged result of a group of input pixels. The pix_per_pax_row, represents the row (input) pixels averaged together to make a single output paxel element.

For example, assume we have a 1024 x 1536 input resolution image, and we desire to do mode 4 averaging (16 row pixels x 16 column pixels = 256 input pixels averaged together for 1 output paxel). The pixel_per_pax_row value would be 16 in this case. This value may also be determined by dividing the input row resolution by the output row resolution (ie 1024/64 = 16).

Theoretically, the S/W could support non-square averaging with minimal modifications. For example, suppose we defined mode 5, which was to average 8 x 16 input pixels and used the same image size as in the previous example. In this example the pix_per_pax_row would be 8 (ie 1024/128 = 8), while the pix_per_pax_col would be 16 (ie 1536/96 = 16), and therefore, not equal to each other. This may be of interest in scanning systems where on CCD binning is done in a single dimension only.

The same data value and ranging assumptions were made with the S/W as the H/W. Basically, 12-bit unsigned data is supported, ranging in value from 0 to 4095 for both input and output values. The S/W masks off any upper values and disregards them. This is consistent with how this was handled in the H/W solution of this thesis.

Memory is efficiently utilized since the input and output data arrays are short data types. This limits them to 16 bits of storage for each data value thus conserving memory usage. When necessary, some data casting is used in the paxelize() function, taking two shorts and casting them to a long. This is done so that long loads and stores may be used to minimize overhead and make the S/W execute as efficiently as possible.

Chapter 5
H/W and S/W Simulation Results

5.1 H/W Testing and Simulation

The averaging ASIC was tested using Esclade Design Book tools, in conjugation with the Model Tech simulator. This tool setup allows for graphical test bench generation of simulation waveforms that provided input stimulus to the design. These input waveforms are easily modified, and simulations could be repeated, as necessary. Output waveforms were examined to insure proper operation.

A hierarchal simulation testing method was utilized to verify the correct operation of the averaging ASIC design. This approach allowed for the lower level components to be tested independently insuring their functionality. After all lower level components were verified; the higher level ASIC chip functionality was tested.

Design Book supports the following three methods for test bench generation: waveform editing, stimulus generators, and expression signals. The waveform generation tool is best utilized for signals that do not have a specific pattern. Twelve stimulus generators are available for generation of repeatable waveform patterns. Boolean expressions may be utilized, when additional signals are desirable to represent the logical combination of several inputs. Utilizing these input methods, a test bench is automatically generated, allowing the application of input waveform vectors to the design, so that the output response may be verified.

Waveform editing and stimulus generators were utilized most often for testing the averaging ASIC design. The waveform editor was utilized for generation of non-cyclic controlling input signals. Input signals, which were asserted on rare occasions, such as horizontal, and vertical sync pulses, and the data available inputs, were generated using this method.

Design Book has an extensive set of stimulus pattern generator functions. For singular control signals they support useful features such as clock and pulse generation functions. For busses, they support counting pattern generators, range of values generator, random number generator, walking ones pattern generator, and walking zeros. The counting generator was most useful for defining input data patterns into the averaging ASIC.

Design book also had several useful tools and methods that aided in debugging the design. "Probe signals" are easily attached to internal node points in the design, and displayed with the simulation waveforms. Re-running the simulations provided immediate feedback to the behavior of the probed signal line. Another tool that was very useful was the "trace event to source" function. This feature allowed waveform transitions, to be traced back to logical functions. This was especially useful in for debugging the behavior of several of the averaging ASIC state machines.

Lower level component simulation testing revealed a significant simulation modeling error with the averaging ASIC. Higher-level logical components were built from a standard library of "module ware" primitives, provided by Esclade. An error was discovered for the D flip-flop with enable shown in Figure 5-1.

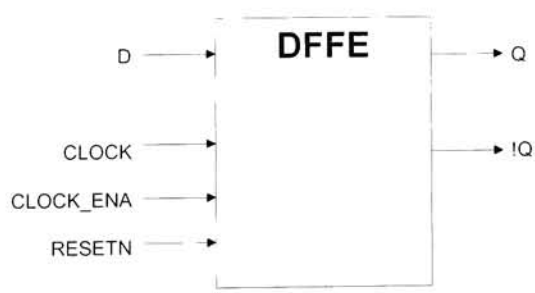

Figure 5-1: D Flip-Flop with Enable Block Diagram

Simulation testing showed that the DFFE component was exiting reset in an undefined state, instead of the output being in a cleared state. These undefined signals propagated

throughout the entire averaging ASIC design. The vendor was notified of this problem, and concurred that it was a modeling bug. The logic shown in Figure 5-2 was substituted for the DFFE. This logic uses a D flip-flop along with a 2:1 mux, to perform the identical function. This also resolved the simulation modeling error while exiting the reset condition.

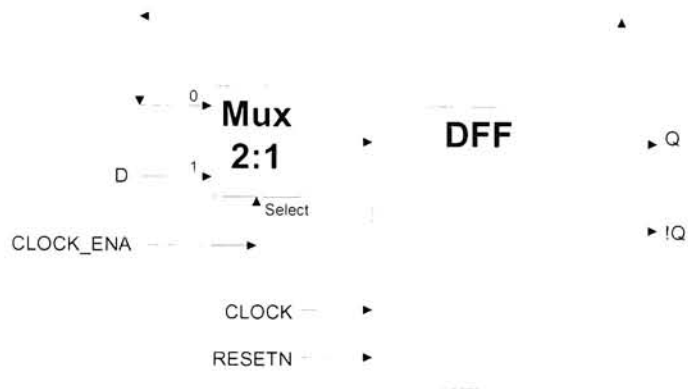

Figure 5-2: D Flip-Flop with 2:1 Mux on Input

Chip level simulation testing was performed to validate functionality of the no averaging and 2x2 averaging modes. First the no averaging mode testing allowed for validation of the data pipeline. Truncated lines of counting pixel data patterns were helpful in debugging the no averaging mode. Simulations were kept shorter and more manageable, which worked out well.

The simulation of the 2x2-averaging mode, functionally verified the overall control of the internal looping back to the B port of the ALU. As mentioned previously, the "trace event to source" function was instrumental in debugging state machine operation, which control the re-circulation of data.

The graphical waveform test bench generation tool worked well for my testing but has limitations. It is a good and efficient match for doing simple functional simulation runs. It allows one to generate stimulus patterns very quickly. Unfortunately this approach is not

practical for an extremely large test vector generation set. If all averaging modes of this ASIC were tested with actual line lengths, this testing would be impractical. It would be necessary to utilize VHDL to define these more rigorous test benches.

5.1.1 Device Synthesis

VHDL files that were output from Esclade, were imported into the Synplicity Synplify place and route synthesis tool. This allowed several vendors silicon such as Altera, Xilinx, or Actel devices to be evaluated. Differing device family technologies among a vendor could also be tested.

The averaging ASIC with its large data path fits best in vendor technology that is register rich. Target devices from the Altera FLEX 10K, FLEX 10KE and ACEX 1K families were determined to meet these criteria. These SRAM based device families are register rich, and lend themselves to this type of data pipeline application. These device families have look-up table architectures, and work well for applications with limited decoding needs. Additional routing delays are often necessary for large decoding functions that require expander cells. Decoding functions for the averaging ASIC were minimal.

Another attractive concept that was considered but not pursued was that these device families have embedded memory blocks. It was felt that these memory blocks could possibly substitute for the discrete input and/or intermediate FIFO memory devices. For systems where circuit board space was critical, this is another advantage for these families. These devices have between 25,000 and 41,000 bits of RAM storage.

Another implementation variation that was considered was the concept of multiple averaging functions or other logical circuitry residing within the same physical programmable device. Systems that desire multiple output resolutions for a given input, a designer could find two implementations of the averaging ASIC logic in a single chip an attractive solution.

Synplify compile time error checking examines the VHDL code, and provides warnings for several "potential design flaw" conditions. The tool will specifically detect and identify cases where the HDL description may synthesize to a transparent latch. Warnings are also provided for cases where signals are input to a block, but are not utilized.

The Synplify complier shown in Figure 5-3 has input settings allowing design constraints and device target technology to be specified. A frequency constraint of 100MHz operation was entered as the desired frequency of operation. EDIF netlist output files were generated for the Flex 10K, FLEX 10KE, and the ACEX 1K devices.

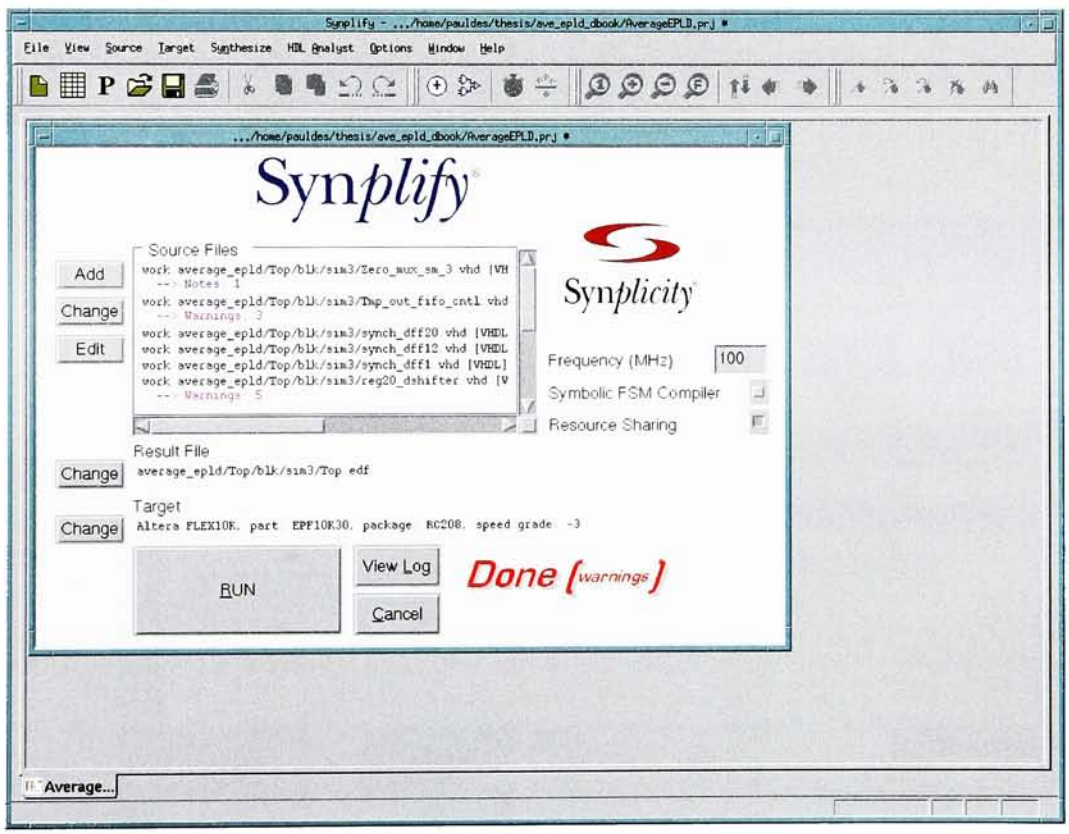

Figure 5-3: Synplicity Synplify Device Synthesis Tool Menu

Synplify generates output reports that indicate the approximate resource utilization and maximum operating frequency. These reports lack the accuracy of those generated by the

place-and-route tool. The next section will detail the results from using the Altera native place-and-route tools. These tools take into account the actual device physical delays, due to the internal silicon routing in the ASIC.

5.1.2 Maximum Clocking Frequency, Chip Utilization, Placement & Routing

Altera Max+PLUS II tools shown in Figure 5-4 below were utilized for post synthesis place and route. The EDIF netlist file created by Synplify is input to this tool, and mapped to the target device technology family during compilation. Max+PLUS II creates a text based report file that more accurately indicates I/O and logic cell utilization for the targeted device. In addition, Altera provides a post layout register performance tool as shown in Figure 5-5. This tool indicates a maximum clocking frequency, taking into account any place and route delays.

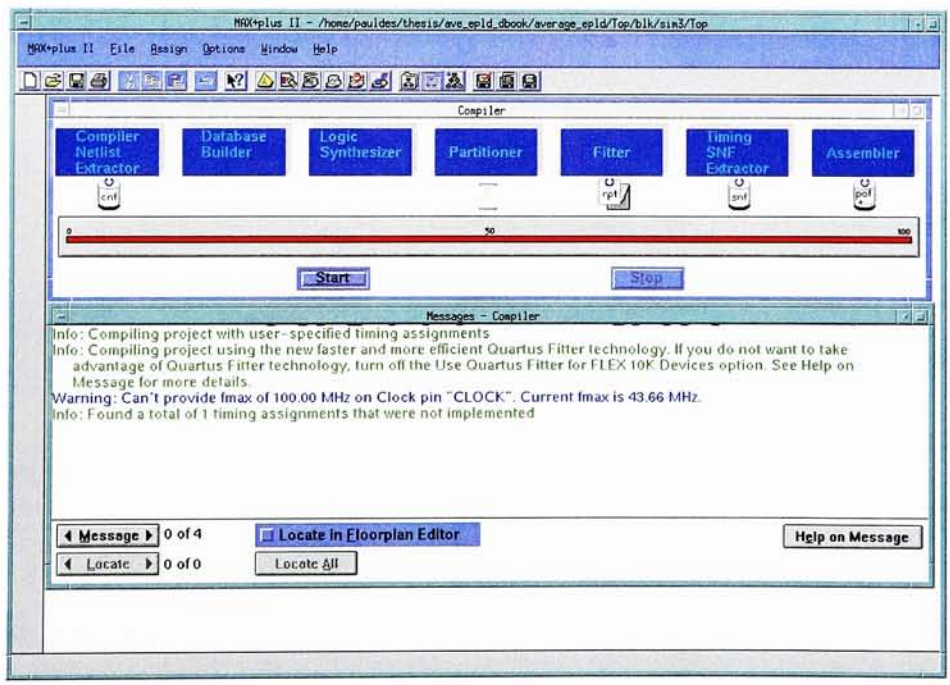

Figure 5-4: Altera MAX+plus II Tool Menu

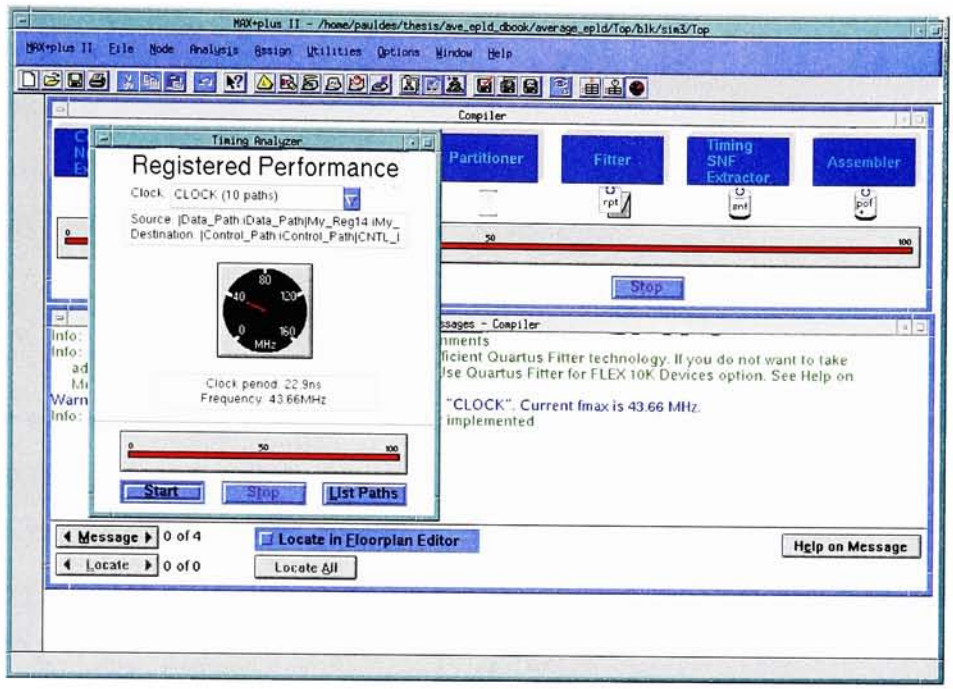

Figure 5-5: Altera Registered Performance Monitor Tool

Table 5-1.shows the Altera place and route results for multiple Altera device types.

ASIC TYPE ->	FLEX 10K EPF10K30QC208	FLEX 10KE EPF10K30EQC208	ACEX 1K EP1K30QC208
Fmax (MHz)	43.66	120	119
Logic Cell Utilization	20%	20%	19%
I/O Utilization	82/102	82/102	82/102

Table 5-1: Place and Route Results for Multiple Devices

The relatively high maximum clocking frequency value was positively impacted by two important factors. First, this design is largely register based pipeline functions. Minimal combinational control and decoding logic functions are utilized. Secondly, the complier has minimal constraints placed on it. None of the I/O pins of logic cells have been locked down.

This allows the complier to have the greatest degree of flexibility in finding a device routing that meets the constraint conditions. Significant logic design changes would be more difficult to incorporate, if the I/O and logic cell placement had been locked down. This would have reduced the maximum clocking frequency.

Table 5-2 indicates averaging ASIC processing times for several devices at multiple input image resolutions. In this table, the holdup processing time refers to the amount of time it would take to process the image, if the entire image was written to the input FIFO, prior to run mode being asserted. With Run_Mode always asserted, the productivity of this circuit is constant, regardless of output averaging mode. An averaged output image will be available at the output of the averaging ASIC, six-clock cycles after the last pixel being read from the input FIFO. The calculations in Table 5-2 assume standard clock oscillator frequencies, plus allowing for some de-rating from the maximum allowable operational frequency.

Horizontal Pixel Resolution	Vertical Pixel Resolution	Altera ASIC Device Type	Clock Frequency (MHz)	Color Planes	"Worst Case" Holdup Processing Time (m Sec.)	"Actual delay" latency from availability of last input pixel (n sec)
1024	1536	EPF10K30QC208	40	3	117.96	150
1024	1536	EPF10K30EQC208	100	3	47.19	60
1024	1536	EP1K30QC208	100	3	47.19	60
2048	3072	EPF10K30QC208	40	3	471.86	150
2048	3072	EPF10K30EQC208	100	3	188.74	60
2048	3072	EP1K30QC208	100	3	188.74	60

Table 5-2: Averaging ASIC Processing Times for Differing Devices and Input Resolutions

The averaging ASIC operating at these frequencies eliminates the processing bottleneck, allowing for real time averaging. Today's systems are likely limited by upstream or downstream data rates. The maximum clocking rate of CCD devices A/D integrated circuits are currently in the 50 MHz range. The ability of the downstream host controller to accept the data limits the overall system productivity. As future technological advances increase the speeds of adjacent circuitry, approaches utilizing an averaging ASIC will enable these systems.

5.1.3 H/W Problems Encountered

The DFFE moduleware simulation problem detailed in the previous section were the only difficulties encountered with the Averaging ASIC design, development and simulation. Esclade design tools worked very well for this application.

Initially, an Altera EPF10K10QC-208 device was targeted. This "10K10"device lacked the logic cell capacity necessary for this design. Unfortunately, the logic cell utilization was over 100%. Maximum operating frequency and I/O performance were in line with the EPF10K30 indicated in the table above.

The "10K30" ASIC device was next larger part that was available in all three device technology families (Altera FLEX 10K, FLEX 10KE, and ACEX 1K). Once mapped to these larger devices, there were no difficulties in meeting performance objectives for operational frequency, I/O or logic cell utilization.

5.2 S/W Simulation & Benchmark Results

The S/W benchmark data was gathered on a 1.0 GHz Pentium III, running the Windows 2000 operating system. This PC was configured with a 133MHz front side bus with 512 MB of Synchronous Dynamic RAM. No other S/W applications were active during the benchmark testing period. The PCI bus productivity is estimated to be 80 MB/sec (32-bits, 33 MHz).

The benchmark code was developed using Microsoft Visual C++ version 6.0 S/W tools. Release versions of this two dimensional averaging S/W were built and executed from the local hard drive. Figure 5-6 shows the user interface displayed upon program execution.

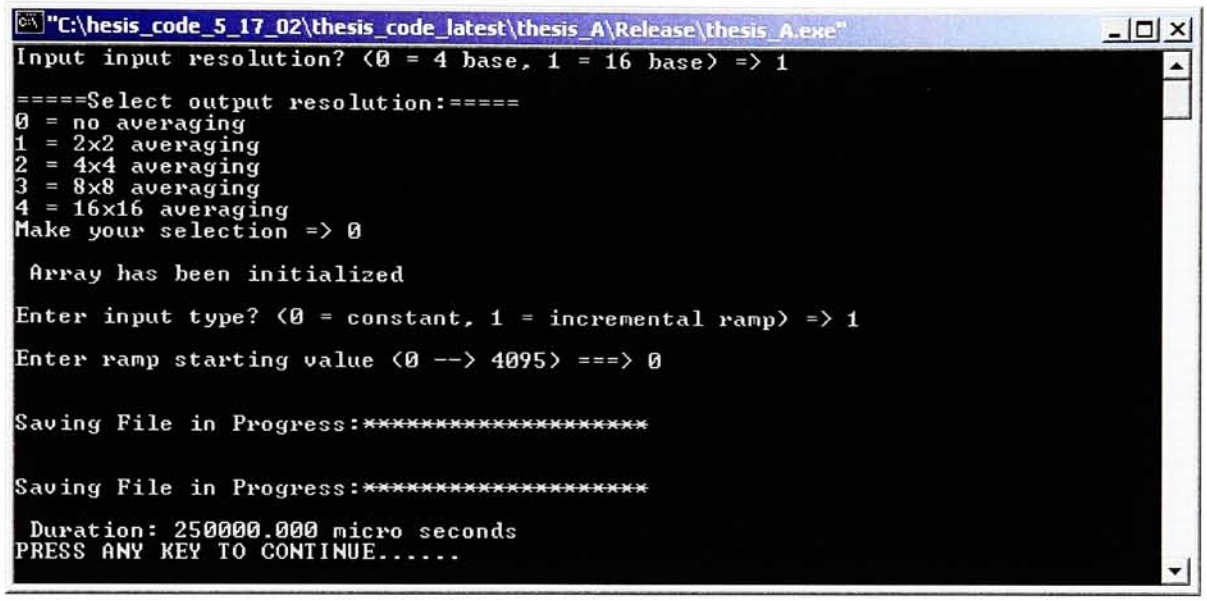

Figure 5-6: Pixel Averaging S/W Simulator User Interface

Statistical samplings of execution times were recorded by running each test scenario 7 times. The fastest and slowest measured values were eliminated, and the 5 remaining measured execution times were averaged. Table 5-3 show the mean execution time for averaging a 2048 x 3072 input image, at various output resolutions.

AVERAGING MODE	MEAN EXECUTION TIME FOR AVERAGING 3 COLOR PLANES (MILLI-SEC.)	PCI BUS TRANSFER TIME (MILLI-SEC.)	TOTAL TIME (MILLI-SEC.)
0	770	0	770
1	345	355	700
2	247	442.5	689.5
3	222	465	687
4	180	471.6	651.6

Table 5-3: Paxelize Execution Times for 2048 x 3072 at Several Output Resolutions

Included in this table is the additional PCI bus transfer time field. This field represents the additional time required to transfer this image to memory utilizing the image reduction size field found in Table 4-2. In the S/W implementation, PCI bus traffic is increased by the size indicated. This additional time would not be present when utilizing the averaging ASIC H/W implementation.

Table 5-4 shows the execution time for averaging a 1024 x 1536 input image, at various output resolutions. Similarly, this table also includes the additional PCI bus transfer time as indicated in Table 4-3, since a reduction in data was not realized. This is the time necessary to transfer the additional data into memory.

AVERAGING MODE	EXECUTION TIME FOR AVERAGING 3 COLOR PLANES (MILLI-SEC.)	PCI BUS TRANSFER TIME (MILLI-SEC.)	TOTAL TIME (MILLI-SEC.)
0	270	0	90
1	150	87.5	137.5
2	120	110	150
3	120	115.6	155.6
4	120	117	157

Table 5-4: Paxelize Execution Times for 1024 x 1536 at Several Output Resolutions

5.2.1 Key Parameters Which Impact S/W Performance

As observed from the indicated execution times, movement of data was the biggest factor that impacted performance. The additional arithmetic operations involved in averaging a larger area were not significant. The timesavings in generating a smaller output image were substantial. For a 2048 x 3072 input image size, the execution time reduced from 770 ns for no averaging, to 180 ns for mode 4 (area of 16 x 16) averaging. For a 1024 x 1536 input image size, the execution time reduced from 270 ns for no averaging, to 120 ns for mode 4 (area of 16 x 16) averaging.

The S/W data averaging approach could also significantly impact PCI bus utilization. The S/W approach utilizes a significantly greater portion of this shared resource. Depending on timing prioritization, and other PCI loads this may or may not be acceptable.

Figure 5-7 shows the Windows 2000 Task Manager performance indicator for three successive runs of the pixel-averaging program. Each of runs consistently utilized 100% of the available CPU time, and approximately 50% of the RAM resources. It may be possible that this could cause some system instability if other process are locked out for a significant time period, or competing for shared resources.

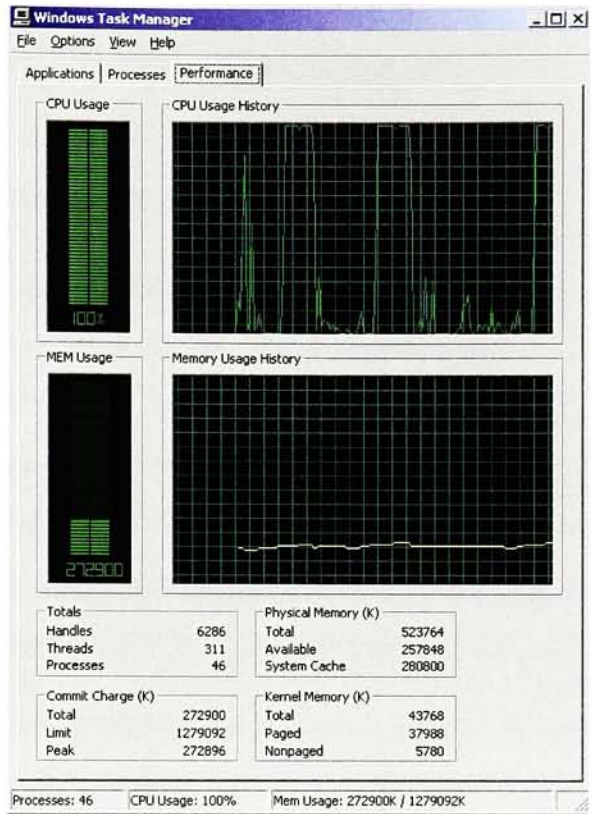

Figure 5-7: Windows 2000 Task Manager - CPU Utilization During Program Execution

5.2.2 S/W Problems Encountered

There were some difficulties in the development of the S/W implementation for two-dimensional pixel averaging. Initially, it was thought that the large two dimensional array sizes were leading to memory allocation problems. Executing the program with input resolutions of 2048 x 3072 resulted in a program crash. It was later discovered that a programming error was the cause of this problem. Mistakenly, the number of rows and column parameters were being swapped. This caused the two dimensional array to be accidentally addressed outside of the memory space boundary for the averaging program.

Chapter 6
Conclusion

6.1 Key Results & Observations

With the widespread utilization of charge-coupled-devices, there is much interest in methods to efficiently process images. The complexity of this task is further increased for systems that require images to be output at multiple resolutions. Designers are forced to consider data reduction techniques, such as the two-dimensional pixel averaging as a tool to complement other techniques, such as sub-sampling or compression algorithms.

This thesis implemented the two-dimensional, pixel data-averaging method for data reduction in H/W and S/W.

A H/W design implementation is presented for two-dimensional averaging of color interleaved pixel data. The design removes a performance bottleneck by enabling real-time processing of images. In introduces only a six-clock cycle latency delay of the data regardless of processing mode. Three programmable devices were identified and operated at a frequency range of 40 – 100Mhz. The design is modular, in that it can be used multiple times for systems that require multiple output resolutions, simultaneously. As future technological advances increase the speeds of adjacent circuitry, approaches utilizing an averaging ASIC will enable these systems to be implemented.

A S/W design is also presented for two-dimensional averaging of pixel data. The S/W testing benchmark results indicates that in several systems this may be practical for systems with lower productivity requirements. Benchmark testing of this algorithm, found that less than 770 microseconds was required to render an image for any supported processing mode.

The S/W solution presented does lack some of the flexibility, modularity and scalability of the H/W solution. Additionally, the S/W approach increases the loading on shared system resources such as the CPU, memory and the PCI bus.

Image quality and compression technologies are becoming increasingly important in today's imaging systems. Protocols will continue to be developed to support applications today and the future. The two-dimensional pixel-averaging algorithm complements other methods. This algorithm implemented in either H/W or S/W efficiently modifies an input image to a lower output resolution, at a quality level superior to that of data sub sampling.

6.2 Suggestions for Improvement & Future Study Opportunities

The H/W and S/W work identified on this thesis could be expanded further.

The averaging ASIC H/W data path and control logic could be modified to include support for the processing of color planar data. An additional "data type" input signal, both pixel interleaved and color planar data could be processed. Additionally, the averaging logic could be included with other image processing algorithms, within the physical device. This could lead to a system on a programmable chip solution.

The S/W algorithms may be expanded to actually average a specified image from a user specified input file, with very little modification. Rewriting the code to support multi-threading may demonstrate significantly different benchmark timing results. This S/W could also be run on different machine types. This could include other desktop systems as well as embedded processors (reduced instruction set computers and digital signal processing devices). Porting the S/W to a similarly configured system, running the LINUX operating systems would provide additional feedback on operating system efficiencies.

Bibliography

[1] Florida State University - Concepts in Digital Imaging web site, "CCD Scanning Formats" http://micro.magnet.fsu.edu/primer/digitalimaging/concepts/scanningformats.html, image used with permission.

[2] Eastman Kodak Company, "CCD Primer, Charge Coupled Devices Image Sensors - Application Note MTD/PS-218", Rochester, NY May 29, 2001.

[3] Eastman Kodak Company, "Solid State Image Sensor Terminology, Application Note DS00-001", Rochester, NY, December 8th, 1994.

[4] Thomas Carducci & Antonio Ciccarelli, & Brent Kecskemety, Eastman Kodak Company, "Ultra-high Resolution 14,400 pixel color image sensor", Proc. SPIE Vol. 3794, October 1999.

[5] Eastman Kodak Company, "Technical Overview: CCD Technology, Technical Information Bulletin TSB-4131", Rochester, NY, June, 1998.

[6] Santa Barbara Instruments Group web page, "About CCD Technology" http://www.sbig.com/sbwhtmls/aboutccd.htm

[7] Eastman Kodak Company, "CCD Image Sensor Noise Sources - Application Note MTD/PS-0233", Rochester, NY. August 8th, 2001.

[8] Dalsa Incorporated, "Introduction CCD Technology Primer", Dalsa Europe, October 20th, 2000.

[9] Eastman Kodak Company, "CCD Primer, Conversion of Light (Photons) to Electronic Charge - Application Note MTD/DS-003", Rochester, NY August 5th, 1999.

[10] Roper Scientific, "Keep the Noise Down! Low Noise: An Integral Part of High Performance CCD Camera Systems – Technical Note", Trenton, NJ, http://www.roperscientific.com/pdfs/technotes/snr.pdf 1999.

[11] Apogee Instruments System Web page, http://www.apogee-ccd.com/ccd103.html

[12] CCD Direct web page, "Pixel Binning – What is it?" http://www.ccddirect.com/online-store/scstore/binning.html

[13] S. Welch, "Novel Techniques for the Efficient Reduction of Data Generated by Charge Coupled Devices Detectors", Review of Scientific Instruments Vol. 71, No. 11, Nov. 2000.

[14] T. Vogelsong, J. Zarnowski M. Pace, and T. Zarnowski, "Scientific/Industrial Camera-on-a Chip using Active Column Sensor CMOS imager core", Photon Vision Systems,Cortland, NY. Proceedings of SPIE Vol. 3965 (2000).

[15] M. French, N. Waltham, G. Newton, and R. Wade, "A Single Chip CCD Waveform Generator and Sequencer", CLRC Rutherford Appleton Laboratory. Proceedings of SPIE Vol. 3355 March 2000.

[16] J. Roberts, "Getting the Image Out – by Getting the Image Out: In Camera Data Reduction Enables Bandwidth Hungry Vision Tasks", Advanced Imaging 1997.

[17] Trevor J Preston, "ABC of CCDs: CCD Terminology De-Mystified", http://www.framos.de/pdf_sheets/CCD_MYST.doc , February 5th, 2002.

[18] Eastman Kodak Company, "KAF Series Full Frame CCD Sensors – Binning Mode Operation", Technical Information Bulletin DS-02-009", Rochester, NY, June, 1999.